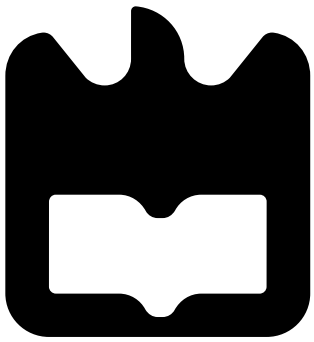
Universidade de Aveiro
Departament of
Electronics, Telecommunications and Computing,
2016

Gabriel Sá Pinto

Ponta de Prova Espectral para IoT

Spectrum Monitoring Probe for IoT



Gabriel Sá Pinto

Spectrum Monitoring Probe for IoT

Dissertation presented to the University of Aveiro for the fulfillment of the necessary requirements to the attainment of the Degree of Master in Electronics and Telecommunications Engineering, performed under the guidance of Nuno Borges de Carvalho, Professor of the Electronics, Telecommunications and Computing Department of the University of Aveiro

the jury

president

Professor Doutor Armando Carlos Domingues da Rocha
Assistant Professor, Universidade de Aveiro

examiners committee

Professor Doutor Henrique Manuel de Castro Faria Salgado
Associate Professor, Universidade do Porto (examiner)

Professor Doutor Nuno Miguel Gonçalves Borges de Carvalho
Full Professor, Universidade de Aveiro (advisor)

acknowledgements

It's with much enjoyment that I'm taking this opportunity to thank to all the ones who've stood beside me during these 5 years. To start off I must thank my advisor, professor Nuno Borges Carvalho, and my co advisor Pedro Cruz, without them this work simply wouldn't happen. I would also like to thank my family and my girlfriend who've supported me in the hardest times. I'd also like to thank to my friends and coleagues who helped me during some of the most important breakthroughs and doubts that came up during this project, namely Fábio Verde, Walter Pires, Henrique Oliveira, Daniel Canedo, Daniel Belo, Daniel Malafaia and Ricardo Correia.

Resumo

O principal objectivo desta dissertação é o desenvolvimento de uma ponta de prova espectral de baixo custo e baixo consumo para a Internet das Coisas que envia os dados recolhidos do espectro para um computador onde serão processados.

No desenvolvimento deste projecto foi usado um Direct Conversion Tuner como front end RF para análise do espectro, seguido de um detector de potência que calcula a potência resultante do sinal de saída I/Q, estes dados são enviados através de um protocolo de comunicação de longa distância implementado em módulos de transmissão e recepção, para um terminal de computador onde serão processados e apresentados ao utilizador em MATLAB.

Abstract

This dissertation's main purpose is the development of a low power and low cost spectrum monitoring probe for the Internet of Things that sends the collected spectrum data to a PC where it will be processed.

In the development of this project, a Direct Conversion Tuner was used as a front end for the spectrum analysis followed by a peak detector to calculate the signal power of the resulting I/Q output signal, these data are then sent through a Long Range protocol used in a pair of Transmitter and Receiver modules, to a computer terminal where they are processed and presented to the user in MATLAB software.

Contents

List of Figures

List of Tables

Acronyms List

3GPP	3rd Generation Partnership Project
ADC	Analogue to Digital Converter
AFC	Automatic Frequency Control
AM	Amplitude Modulation
BBG	Bandbase Gain
BLE	Bluetooth Low Energy
BPF	Band Pass Filter
BPSK	Binary Phase Shift Keying
BW	Bandwidth
CAD	Computer Aided Design
CDMA	Code Division Multiple Access
CRC	Cyclic Redundancy Check
CSS	Chirp Spread Spectrum
DAC	Digital to Analogue Converter
DC	Direct Current
DSP	Digital Signal Processing
DVB-S2	Digital Video Broadcasting - Satellite - Second Generation
ETSI	European Telecommunications Standards Institute
FDD	Frequency Division Duplex
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FSK	Frequency Shift Keying
GFSK	Gaussian Frequency Shift Keying

GMSK	Gaussian Minimum Shift Keying
GND	Ground
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUIDE	Graphical User Interface Development Environment
GUI	Graphical User Interface
I/Q	In Phase / Quadrature
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IF	Intermediate Frequency
IIP3	3rd Order Intercept Point
IMT	International Mobile Telecommunications
ISM	Industrial Scientifical Medical
IT	Institute of Telecommunications
IoT	Internet of Things
LNA	Low Noise Amplifier
LO	Local Oscilator
LPF	Low Pass Filter
LPRD	Low Power Radio Device
LPWAN	Low Power Wide Area Network
LP	Low Power
LTE	Long-Term Evolution
LTN	Low Throughput Network
LoRaWAN	Long Range Wide Area Network
LoRa	Long Range
M2M	Machine to Machine
MCU	Micro Controller Unit

MSK	Minimum Shift Keying
NFC	Near Field Communication
OOK	On-Off Keying
P2P	Peer-to-peer
PA	Power Amplifier
PCB	Printed Circuit Board
PHY	Physical Layer
PLL	Phase Locked Loop
RAN	Radio Access Network
RBW	Resolution Bandwidth
RFID	Radio Frequency Identification
RLC	Radio Localization Service
RF	Radio Frequency
RMS	Root Mean Square
RPMA	Random Phase Multiple Access
RSSI	Received Signal Strength Indicator
RVA	Aeronautics Radionavigation Service
RX	Receiver
SA	Spectrum Analyzer
SDR	Software Defined Radio
SNR	Signal to Noise Ratio
TDD	Time Division Duplex
TETRA	Terrestrial Trunked Radio
TQFN	Thin Quad Flat No Lead
TTL	Transistor Transistor Logic
TX	Transmiter
UART	Universal Asynchronous Receiver Transmitter
UMTS	Universal Mobile Telecommunications System
UNB	Ultra Narrow Band

USB Universal Serial Bus

VAS VCO Autoselection

VCO Voltage Controlled Oscilator

VGC1 Voltage Gain Control 1

VSAT Very Small Aperture Terminal

VSA Vector Signal Analyzer

Chapter 1

Introduction

1.1 Context and Motivation

Nowadays, there is an emerging quantity of protocols and modulation standards. The electromagnetic spectrum distribution table [1] can give us an idea of how immense and diverse the ways of communicating with devices and with the world surrounding us really are. The spectrum bands are getting more and more bogged and it's becoming really hard to allocate free bands for new standards or protocols, thus being necessary to monitor the spectrum and be able to understand it.

Another aspect that conditions the spectrum is the constant evolution of the complexity associated to each protocol. Let's take the example of mobile networks [2], once there was 1G and now we're all the way up to 4G with 5G already in development, counting the fact that each generation had to support several standards, therefore enhancing the complexity of the protocols and the systems themselves. This leads to the need of having an SDR capable of interpreting all of those protocols and that has the ability to reconfigure itself and adapt depending on the service. However, these radios cannot be fully implemented yet due to physical limitations, until then we can only contribute with systems based on this SDR concept.

A Spectrum Analyzer is a device that can basically analyze the spectral composition of a signal at a certain frequency. Applying the radio concept to Spectrum Analyzers we can make it portable and even a Spectrum sensor that can send the acquired data to a database.

The Internet of Things is a concept which implies a connection between electronic devices, not only to the users themselves but with the world wide web as well. The aim nowadays is to get as much functionality as possible from our phones, gadgets, and even from our own homes, as we want to know what's going on and even control them in real time. This is a concept that has been growing at a huge rate on the latest years and is bound to be the future of modern technology.

> *"The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so."*
>
> — Kevin Ashton, Atmel

1.2 Objectives

The main goal of this dissertation is to develop a system, based on SDR architectures, for the monitoring of the Electromagnetic Spectrum comprising the biggest possible portion of frequencies between 30MHz and 6GHz, since these frequencies contain the most relevant communication protocols in wireless transmissions, namely: TV; TETRA; mobile communications (GSM, 3G-UMTS, 4G-LTE, Wi-Fi; IoT technologies).

This Spectrum Probe is supposed to be a low cost and low power sensor capable of transmitting the spectrum information, through a long range communication protocol, to a PC where the information will be treated, analyzed and presented in MATLAB software, which will be the interface with the user.

1.3 Structure

This thesis is organized in the following way:

- **Chapter 2**: A theoretical introduction to the concept and architectures of a Spectrum Analyzer, followed by a deeper understanding on the concept of Software Defined Radio.
- **Chapter 3**: A short briefing on the Internet of Things, its applications and the most relevant communication protocols.
- **Chapter 4**: Describes the project of the whole system, namely the architecture and the approach used in the project of the spectrum analyzer; the microcontroller's functions and needed communication protocols; the choice for the long range wireless communication modules.

  The second part explains the implementation to all the programing and algorithms needed for the long range communication modules, the spectrum analyzer chip and MATLAB data processing.
- **Chapter 5**: Presents the results of the tests performed with the communication modules and the spectrum analyzer chip.
- **Chapter 6**: Concludes the body of this dissertation and summarizes all the work done as well as presenting some future improvements and features that can be added to the system.

Chapter 2

Spectrum Analyzers and SDR

2.1 The Spectrum

It makes sense to start this work at its core, so what actually is the electromagnetic spectrum? Nothing but electromagnetic energy travelling in the form of waves which is spanned in a long spectrum from long radio waves to short gamma rays. However, the human eye can only recognize a small portion of this so called spectrum, that is the visible light.

In mathematical terms, from [3] and [4], we know that the spectrum is the representation of a signal (a sine wave for example) in the frequency domain. The way the frequency domain relates to the usual time domain is through the Fourier transform, with that said we have these two representations:

In the frequency domain:

$$X_f(f) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t).e^{-j2\pi ft}.\, dt \tag{1}$$

Or in the time domain:

$$x(t) = F^{-1}\{X_f(f)\} = \int_{-\infty}^{\infty} X_f(f).e^{j2\pi ft}.\, df \tag{2}$$

Where:

$\{X_f(f)\}$ is the complex signal in the frequency domain.
x(t) is the same signal but in the time domain.
$F\{x(t)\}$ is the Fourier transform of x(t).
$F^{-1}\{X_f(f)\}$ is the Inverse Fourier transform of $F\{x(t)\}$.

The way to calculate a spectrum of a given signal depends on whether the signal is periodic or not.

For periodic signals in the time domain we can use the Fourier Series which is the resulting

sum of sine and cosine signals of different frequencies and amplitudes, and it is described as:

$$x(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \,.\, sin(n.2\pi f_0.t) + \sum_{n=1}^{\infty} B_n \,.\, cos(n.2\pi f_0.t) \tag{3}$$

Where A_0 represents the signal's DC component, A_n and B_n the coefficients of the signal at each frequency component. These can be calculated through equations (5) and (6) using the Fourier Transform in equation (1) with the respective periodic intervals.

Applying the Fourier Transform will result in:

$$A_0 = \frac{2}{T_0} \int_0^{T_0} x(t) \,.\, dt \tag{4}$$

$$A_n = \frac{2}{T_0} \int_0^{T_0} x(t) \,.\, sin(n.2\pi f_0.t) \,.\, dt \tag{5}$$

$$B_n = \frac{2}{T_0} \int_0^{T_0} x(t) \,.\, cos(n.2\pi f_0.t) \,.\, dt \tag{6}$$

In practice, the frequency spectra of the sine and cosine signals can only be represented through discrete spectral lines with certain amplitudes represented by δ as presented next in equations (7) and (8):

$$F\{sin(2\pi f_0.t)\} = \frac{1}{2j}(\delta(f - f_0) - \delta(f + f_0)) \tag{7}$$

$$F\{cos(2\pi f_0.t)\} = \frac{1}{2j}(\delta(f - f_0) + \delta(f + f_0)) \tag{8}$$

Thus it's possible to decompose any periodic signal into harmonic related coefficients by using Fourier decomposition. An example can be this decomposition of a rectangular signal in figure 1.

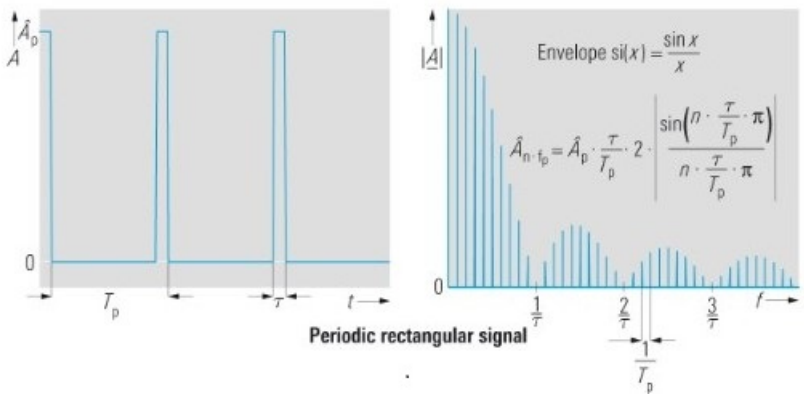


Figure 1: Decomposition of a periodic signal [3]

In the case of non-periodical signals, they cannot be represented by a Fourier series, since they possess no discrete spectral components but they have a continuous behavior

spanning the whole spectrum. However, the Fourier Transform can be applied in order to provide knowledge of the occupancy. The figure below shows the decomposition of non-periodic signals, in a) we have the decomposition of Gaussian noise and the below in b) the decomposition of a regular I/Q telecommunication signal.

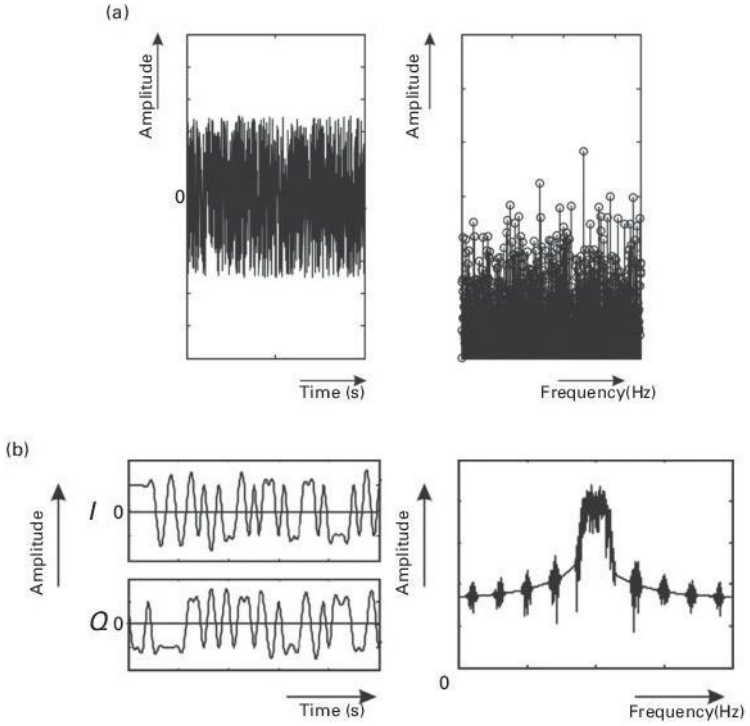


Figure 2: Decomposition of Gaussian noise (a) and I/Q signal (b) [4]

This is why spectrum analyzers are so important, as they can present the spectrum in a given window. The next section will discuss the different types of spectrum analyzers.

2.2 Types of Spectrum Analyzers

A Spectrum Analyzer (SA) is an essential instrument for engineers, providing a view of the spectrum of signals with their amplitudes and frequencies. They are widely used within the electronics industry, mainly for analyzing the radio frequency spectrum. [5]

There are many types of spectrum analyzers, the ones that stand out will be presented in the next subsection.

2.2.1 FFT Analyzers

These are the type of SA's that use the FFT, in other words, they calculate the frequency spectrum from a signal captured in the time domain. In theory the analyzer would have to apply equation 1 (FFT) to calculate the spectrum in a period of infinite length, which is obviously, not possible in practice. However, it is possible to determine the signal spectrum with sufficient accuracy. The following block diagram shows the typical architecture of an FFT analyzer.

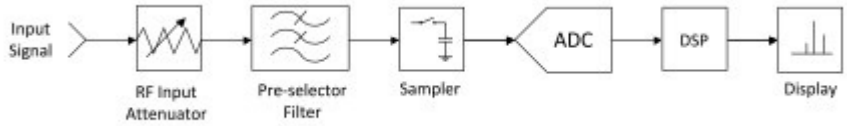


Figure 3: Simplified block diagram of an FFT analyzer [6]

The blocks can be explained as follows:

- **Variable Gain / Attenuators front end**: These are needed to ensure the signal is at the right level for the Analogue to Digital converter range, thus optimizing its performance and resolution.

- **Low pass filter**: This filter serves as an anti – aliasing filter, which is required because the sampling rate of the system within the FFT is important, that is, the waveform must be sampled at a high enough rate. This will be further explained on the next subsection.

- **Sampler and ADC converter**: As the name states these blocks will convert the wave into a digital form. It does it by sampling time intervals and outputting a digital format required for the FFT analysis.

- **DSP**: This block will deal with all the processing of the signal regarding the digital domain.

- **Display**: After proper processing of the data that outputs from the FFT analyzer the information can be presented in the most suitable way to its user.

How an FFT analyzer works

Now, one must proceed to explain how the sampling system works, as referred in the block explanation. To ensure that aliasing effects do not inflict error during signal sampling, it's necessary to limit the bandwidth of the input time signal. Aliasing, also known as image frequency, is a phenomenon that can be defined as the overlapping of frequency bands. According to Nyquist's theorem in equation 9 the sampling frequency must be greater or equal to twice the maximum input frequency, in order to prevent aliasing.

$$f_s \geq 2 \, . \, f_{in\ max} \tag{9}$$

The following images show a good example of a signal with and without aliasing effects:

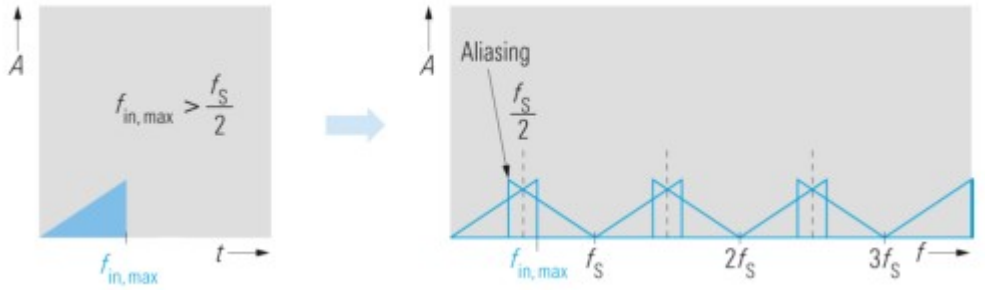


Figure 4: Signal with aliasing effects [3]

$$f_{in\ max} > \frac{f_s}{2} \tag{10}$$

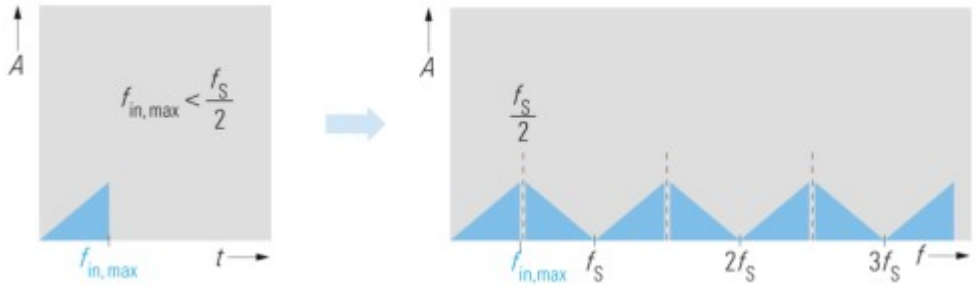


Figure 5: Signal without aliasing effects [3]

$$f_{in\ max} < \frac{f_s}{2} \tag{11}$$

In this kind of analyzers only a portion of the signal is considered for the Fourier transform which is performed during a windowing process.

FFT Analyzer vs Oscilloscope

Also its worth mentioning that FFT analyzers are often confused with **oscilloscopes**, and there are in fact noticeable similarities regarding the sampling of the signal in the time domain, and then offering the option for spectral display. Nonetheless there is a substantial difference between them, such as different criteria being applied when selecting the ADC.

An SA has the distinguishing feature of its high dynamic range, for that effect an ADC with greater quantization depth and lower sampling rate is selected, on the other hand, oscilloscope designers tend to select ADCs with high sampling rate in order to be able to properly display waves in the time domain.

Disadvantages

The main disadvantage of FFT analyzers is its maximum frequency limit imposed by the ADC, which requires very high sampling rates that aren't even available nowadays, leading to the need of down converting the signal to a lower frequency, which imposes bandwidth limits to the SA. Summing up, this architecture requires high performance ADCs which are expensive, so the FFT analyzer usually has very high costs associated.

2.2.2 Heterodyne type SA (Swept-tune method)

This type of SA differentiates from the last one mainly because the input's signal spectrum is not calculated from the time characteristic, but instead, it's calculated by performing an analysis directly in the frequency domain. Although there are more receiver architectures used in this kind of analyzers, here we shall discuss the superheterodyne because it is by far the most used type of spectrum analyzer, and one of the most complete, so to say. As we can tell, by the name, it uses the superheterodyne principle and its operation deals up to very high frequencies. The figure below shows a block diagram of a superheterodyne analyzer.

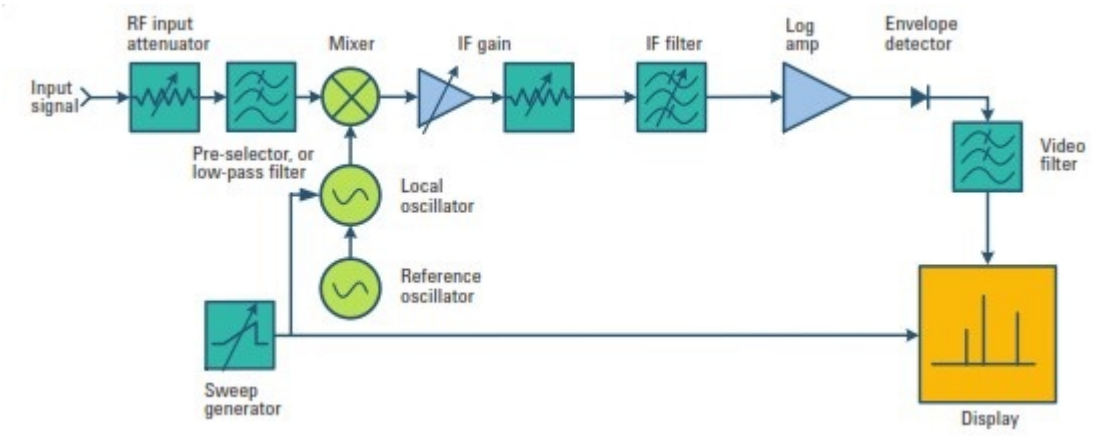


Figure 6: Superheterodyne Architecture [7]

Stages of Operation

This sub section is intended to study the main blocks and their functions inside the figure above. As [1] states, the main blocks are as follows: the RF input attenuator; the mixer and oscillator; the IF amplifier and filter; the envelope detector; video filter; sweep generator and finally the display.

- **Input attenuator**: This is the first stage of the architecture, the signal passes through this attenuator in order to adjust the power level that enters the SA. It will prevent mixer gain compression or distortion due to high level signals. However, upon the use of this attenuator, the noise factor of the overall system will increase. Usually between the input attenuator and the mixer there is also an LPF to prevent high frequency signals from reaching the mixer.

- **Mixer and oscillator**: This stage is the one responsible for the down conversion of the input signal to the IF by mixing the output of the LPF with a signal from the LO. The following equation can describe the output signal frequency:

$$|mf_{LO} \pm nf_{in}| = f_{IF} \tag{12}$$

Where m,n = 1, 2. . . , f_{LO} is the local oscillator frequency, f_{IN} is the input frequency, and f_{IF} is the intermediate frequency. This is also the stage where one of the main disadvantages of the superheterodyne principle comes up, which is the appearance of image frequencies as seen in figure below.

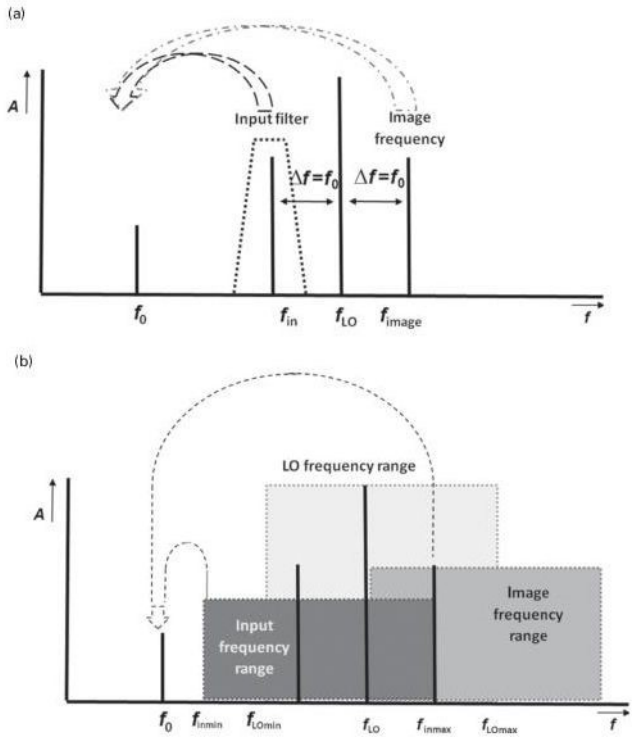


Figure 7: Down conversion of both the signal and its image frequency (a). Covered bandwidth problem(b) [4]

These can and should be eliminated through the use of filters, however the filtering stage can get pretty complex if the SA has to cover several frequencies.

- **IF gain and IF filter**: The IF logarithmic amplifier allows for an adjustment of the vertical position of the signal on the display just before it goes into the BPF. This filter has an adjustable bandwidth known as RBW. Selecting a lower RBW, for instance, results in a better selectivity and consequent lower SNR (as seen in figure 8), but then again it lowers the sweep speed and trace update rate.

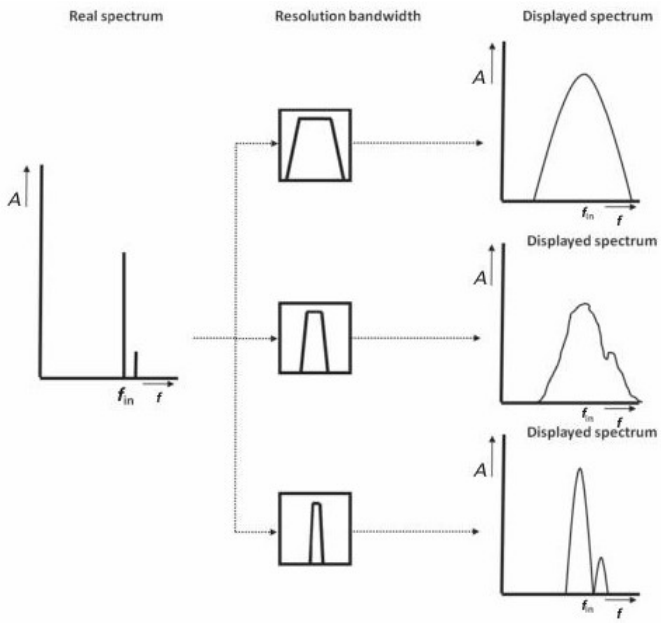


Figure 8: Impact of RBW in the displayed spectrum [4]

- **Envelope detector**: After the filtering the signal proceeds to this block and gets rectified. Usually these rectifiers or detectors are based of diode detectors which basically returns the power of the filtered signal. They can be several types of detectors, such as peak detectors, sample detectors, RMS detectors, or average detectors.
- **Video filter**: The signal enters the last stage before the display, which is an LPF and smoothens the trace to be displayed. In the image below we can see how the envelope detector and video filter work in practice.

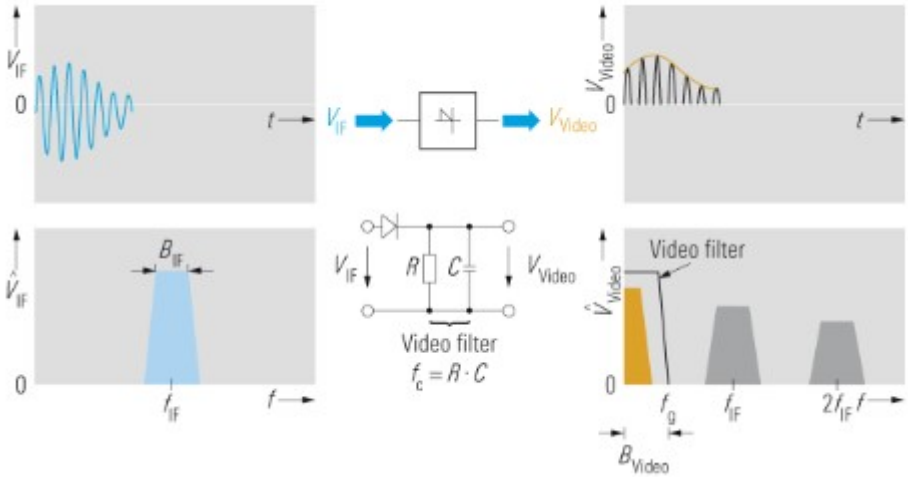


Figure 9: Envelope and video filter demodulation [3]

- **VCO and sweep generator**: Finally, the VCO (Voltage Controlled Oscillator) is tuned by the sweep generator in order to change the frequency proportionally to the ramp voltage of the sweep generator. This frequency is the one being displayed.

This concludes the block diagram analysis. It is important to retain some advantages and disadvantages of the superheterodyne principle.

Although being quite expensive, it's fairly cheaper than the FFT analyzer for the same levels of performance which is why this type of analyzers is often preferred. Of course it has some downsides, and they lie with the inability to measure a signal's phase (only measures amplitude) and it also can't measure transient events effectively, because the process of sweeping takes time. More front end architectures will be approached in subsection 2.3.5.

2.2.3 Vector Signal Analyzers

This is a type of instrumentation that's already considered a great update from the traditional SA. VSA's can be seen as a combination of superheterodyne technology with high speed ADC's and other DSP technologies, which provides for fast, high resolution spectrum measurements, demodulation and advanced time-domain analysis.

One of the characteristics that distinguishes VSA's from regular SA's is that it's basically a digital system using digital data and algorithms to perform data analysis, while normal spectrum analyzers are essentially swept-tuned analog systems. Similarly to the process of a SA, in a VSA, the signal is manipulated in the same way, although, it's digitally converted in an ADC before entering the RBW as we can see in figure 10.

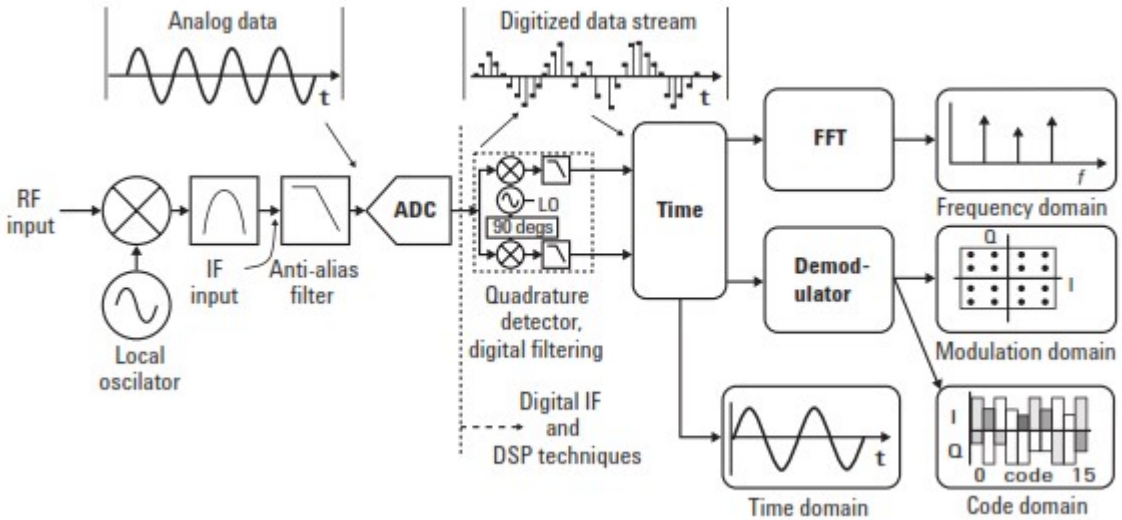


Figure 10: Simplified diagram block of a VSA [8]

These devices are explicitly designed to measure and deal with complex data, becoming a fundamental tool in an RF laboratory, since nowadays signals get progressively more complex.

It also fills within the disadvantage gaps of analog superheterodyne SA's, one of them was already stated and that is the capacity of processing more complex and dynamic signals, another one is that they can measure magnitude and phase of the input signal. Their range is also increased since VSAs can cover RF and microwave ranges and have additional modulation-domain digital capability.

However, there are also some problems that arise that weren't present in SAs, an example can be the dynamic range being limited due to thermal noise, and also the maximum number of bits in the ADC is important, because signals are digitized before the FFT calculation.

2.3 Software Defined Radio (SDR)

2.3.1 Introduction

Since this thesis is based on the concept of an SDR it's logical to dig a little deeper into this subject.

As stated in the context section of this thesis, there has been an exponential growth in the ways and means by which people communicate, that includes data, voice and video communications, as well as broadcasting, emergency response communications, and so on. However, modifying and redesigning radio devices cost-effectively is crucial in these situations. From this need arises the concept of SDR.

This concept first came up in the early 90s, when Joseph Mitola introduced and described SDR as:

> *"A software radio is a radio whose channel modulation waveforms are defined in software. That is, waveforms are generated as sampled digital signals, converted from digital to analog via a wideband DAC and then possibly upconverted from IF to RF. The receiver, similarly, employs a wideband ADC that captures all of the channels of the software radio node. The receiver then extracts, downconverts and demodulates the channel waveform using software on a general purpose processor."* [9]
>
> — Joseph Mitola

In other words, Mitola defines an ideal SDR as a radio that implements its radio interface by software and reconfigures itself based on the medium they find themselves in.

The model proposed by Mitola, for an ideal SDR is presented on figure 11. Its front end is composed by and only an ADC and a DAC that make the respective conversions, as for the rest of the components like filters and mixers, they are all implemented via software in the DSP. This should provide the system with flexibility since the only thing that really changes is its software.

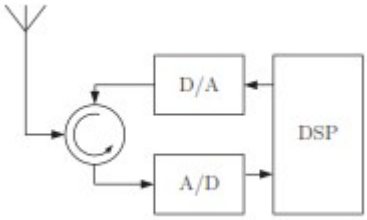


Figure 11: Ideal SDR concept by Mitola [10]

For practical reasons, this ideal SDR is not possible to achieve in the current days, nor will it be possible anytime soon due to the state of the art of the intervening ADC, DAC and DSP, since these impose limitations on computational capacity of contemporary processors and also have huge power consumptions.

However, an SDR doesn't have to be "ideal", thus the fact that one can't possibly define the term "ideal" in this case, not knowing any consensus on the level of reconfiguring capacity needed to qualify it as such. What's important to retain is that an SDR is a radio where the physical layer is essentially defined by software.

One of the first implementations of SDR systems was made by the US military in the 1970s, way before Mitola came up with the concept of SDR, this military project was known as SPEAKeasy. SPEAKeasy is a multi-phase, joint service technology program to prove the concept of a programmable waveform, multiband, multimode radio. Although this was a portable device, it wasn't a handheld radio.

Figure 12: Air-to-Ground Radio Transceiver Unit [11]

After this implementation, there were other known attempts such as the JTRS (Joint Tactical Radio System) based on Speakeasy's technology, which was presented in 1998. Then came some commercial SDR systems, one example can be seen in figure 12.

2.3.2 SDR Architectures

One of the key points of SDR is digitization, and since an ideal SDR as presented on figure 11 cannot be implemented nowadays, different architectures were studied to achieve a similar effect based on this key point. Thus came three main architectures based on the digitization point in the reception chain, them being: Base Band Digitization, IF Digitization and RF Digitization.

Base Band Digitization

In this architecture, demonstrated in figure 13 the ADC is located in the baseband. The signal must pass through two stages of down conversion with filtering and then it's amplified by an LNA with the intent of shifting the signal to baseband.

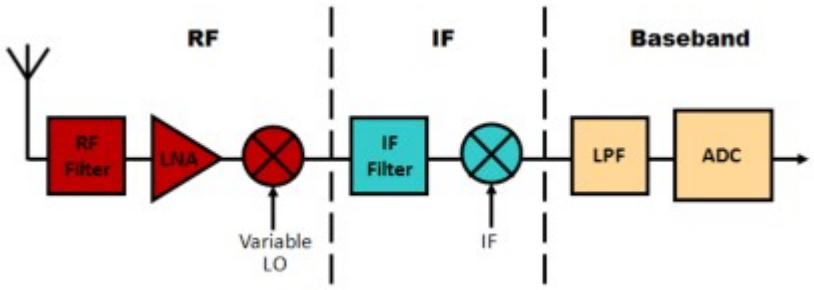


Figure 13: Base Band Digitization Architecture [12]

This technique is commonly used among radio receivers, and it benefits from having low cost narrowband RF and IF components with also low power consumption, yet this also causes huge limitations for SDR applications, precisely because SDR systems aim at broadband. It also suffers from a high number of components, making it hard to implement it in IC.

IF Digitization

Again, regarding its name, this architecture implements the ADC in the IF stage of the signal chain, preceded by an RF filter, which eliminates out-of-band interferers (figure 14). This makes up for the fault of the previous model broadening the bandwidth, this is possible because after the down conversion stage, the bandwidth is digitized allowing for it to be digitally processed, allowing not only a wider band, but also multiple channel selection.

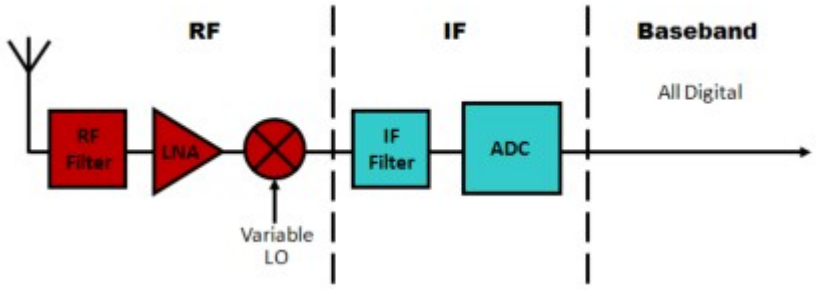


Figure 14: IF Digitization Architecture [12]

Moreover, this architecture allows for a more compact design in IC's, but has however the disadvantage of higher power consumption in its components.

RF Digitization

From all three architectures presented here, this is the one that most resembles an ideal SDR. If we recall the ideal SDR concept by Mitola in figure 11, and take a look at figure 15 we can see the resemblance, having the most phases implemented by software, as possible.

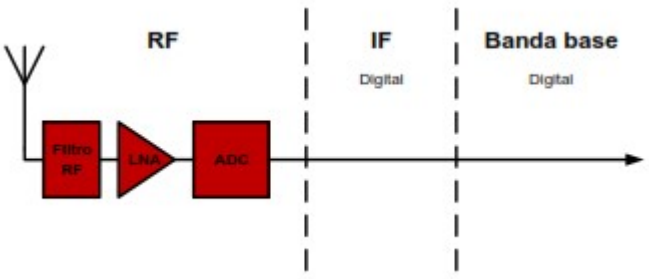


Figure 15: RF Digitization Architecture [12]

Notwithstanding, this architecture presents many challenges, to the nowadays' technology, making it unfeasible, again due the high specs for the ADC. Even though there are already some very high speed ADCs in the market with speeds over 50Gs/s [13], they are still too expensive for commercial applications.

2.3.3 Benefits of SDR technologies

As previously stated, SDR technologies would make everything easier. Let's take a look at three perspectives from [14] envisioning: Radio Equipment Manufacturers; Radio Service Providers and of course the End Users.

For **Radio Equipment Manufactures and System Integrators**:

- Radio products being implemented by a universal platform architecture, allowing for faster production, and market insertion;
- Reusing software in different radio devices, reducing development costs;
- Remote reprogramming, allowing to solve issues and bugs while the radio is in function, reducing costs of operation and maintenance.

For **Radio Service Providers**:

- New features and add-ons to the existing service, without major costs, allowing providers to be ready for future implementations on their networks;
- Common radio platforms for multiple markets, reducing logistical support;
- Remote software downloads, through which capacity and capability upgrades could easily be activated and new function can be added.

For **End Users**:

- Essentially reduce costs of wireless communication services;
- Enable communication with everyone, anywhere and anyhow they need so.

As for the **disadvantages**, it has been previously stated that SDR's main drawback is the ADC component (please check the previous sub section) .

2.3.4 SDR Related Technologies [14]

Adaptive Radio

An adaptive radio is a device in which communication systems have the ability to monitor their own performance, and make the necessary adaptations to improve this performance, thus the term adaptation.

Cognitive Radio

Also developed by Joseph Mitola [15], the concept of cognitive radio can be defined as a radio in which the communication systems are not only aware of their performance and internal state, but also from the environment that surrounds them allowing them to perform a spectrum analysis and identify unoccupied spectrum "slices" in which they can operate. However, this implies the need of a front end capable of covering as many spectrum span as possible.

It's worth mentioning this concept is the core of this dissertation and SA project.

Intelligent Radio

An intelligent radio is a cognitive radio that actually has the ability to learn. This allows for the radio to improve the way it adapts to the surrounding medium and to better serve the end user, making it the peak of advanced wireless technologies.

The way how all these concepts relate within advanced wireless technologies, can be seen in the Venn diagram depicted in figure 16.

Figure 16: Venn Diagram showing the relation between associated advanced wireless technologies [14]

It's also worth mentioning that these concepts don't necessarily describe a single piece of equipment, but may be spread across a network.

2.3.5 RF Front End Receiver Architectures

An RF front is defined as the block located between the antenna and the digital baseband system, since waveforms propagate in the air, an analog front end is needed to receive the radio signals. The receiver area includes all the filters, LNAs, down conversion mixers used to process the modulated signals received at the antenna into signals suitable for digitization in the ADC, that is why sometimes an RF front end is also referred as the " ADC to baseband part" of a receiver.

In this section, some of the most common architectures for RF front ends will be described and compared with each other.

Superheterodyne architecture

This architecture has been previously referred as the most commonly used in RF receivers. In figure 17 a superheterodyne architecture that has I/Q modulation is shown.

This configuration is based in two stages of down conversion, i.e, the signal is first demodulated into IF and then into a baseband signal. The difference between this configuration and the previous one in figure 6 is that this one makes an I/Q modulation in order to achieve better amplitude / phase information from the signal received. Since the principle of operation was already described in section 2.2.2 this sub section will jump right into advantages and disadvantages of this architecture.

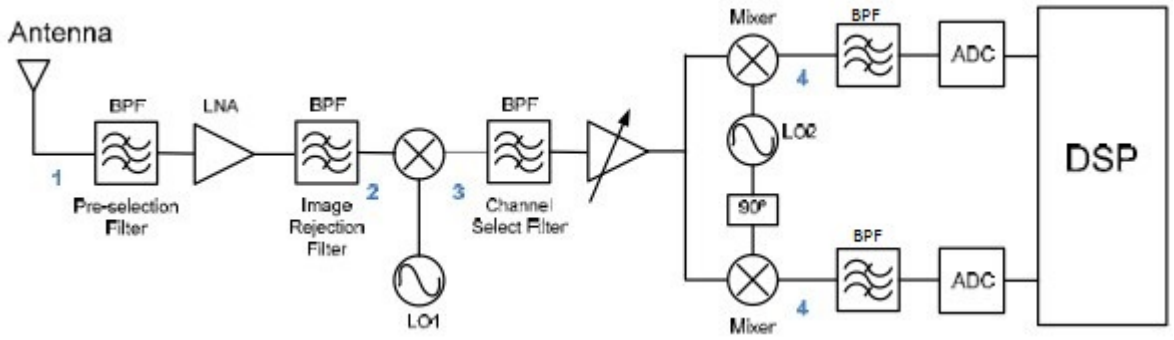


Figure 17: Superheterodyne Architecture [16]

Advantages

The main reason why this configuration is adopted for most radio receivers is the availability of low cost narrowband RF and IF components with low power consumption. Moreover, it provides good levels of sensitivity, that is, allowing lower power signals at the input to have a decent SNR at the output. It also offers selectivity, which is the ability to separate the desired band from signals at other frequencies. At last it is immune to DC offset problems that affect other architectures.

Disadvantages

The most noticeable drawback of this configuration is image frequency. As approached in section 2.2.1 image frequency must be cancelled, and for that purpose this architecture

requires filters composed by high quality discrete components which are not favorable to the modern day's IC technologies. Needless to say, this architecture is also one of the most complex ones.

Despite this architecture being the most used in radio receivers, it's not the most suitable for SDR systems because it has the limitation of not dealing with wideband signals.

Zero-IF Architecture

This architecture is sort of a simplified version of a superheterodyne, because instead of having an IF stage, it converts the RF signal directly to baseband, thus also being known by Direct Conversion Architecture. Because the frequency of the LO is set to the same frequency of the desired input signal, this architecture is further known as a Homodyne receiver.

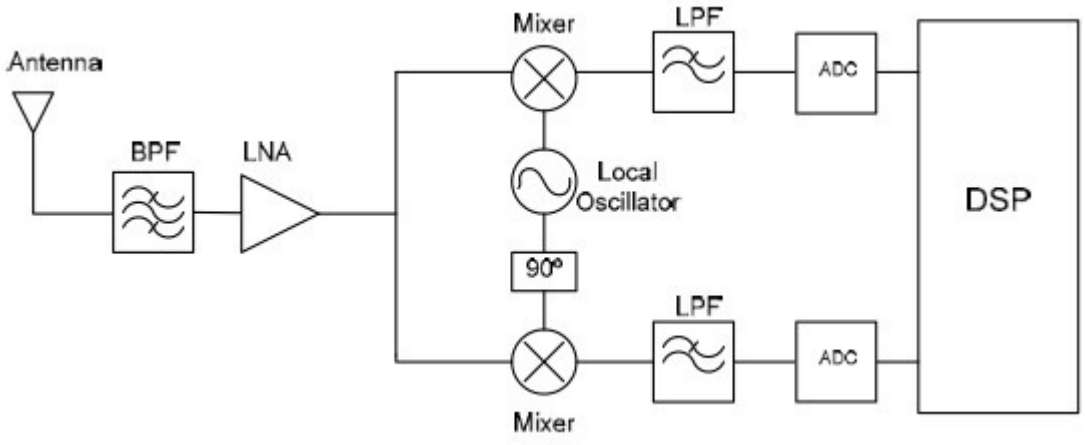


Figure 18: Homodyne Architecture [16]

Figure 18 shows this architecture, the received signal is selected in a BPF and it's amplified by the LNA, then it's down converted to DC by two mixers for I/Q modulation, finally being digitized in the ADC.

Advantages

When compared to the superheterodyne architecture, this one has a visible reduction in analogue components and therefore in complexity as well, guaranteeing a good IC application. Furthermore, this architecture also benefits from an easier image suppression.

Disadvantages

The main problem associated with this configuration is the direct conversion to DC. This section may cause many problems, such as DC offset, LO leakage, due to non-ideal isolation between the port from the mixer, I/Q modulation errors, distortion caused by nonlinear components and flicker noise from the mixer.

Some techniques can be applied to reduce these downsides however they'll always be a strong condition to choose the superheterodyne over the homodyne. Because of these problems associated with the LO and mixer, the components used have to be high quality and therefore more complex, despite the overall simplicity of the architecture.

Even though this configuration allows the reception of wideband signals, which is important for an SDR, the limitations imposed above makes it hard to implement low cost components, since it requires stable LO's and other expensive analog components.

Low-IF Receiver

This configuration, presented in figure 19 although similar to the last one, tries to combine the best of both worlds between the superheterodyne and homodyne.

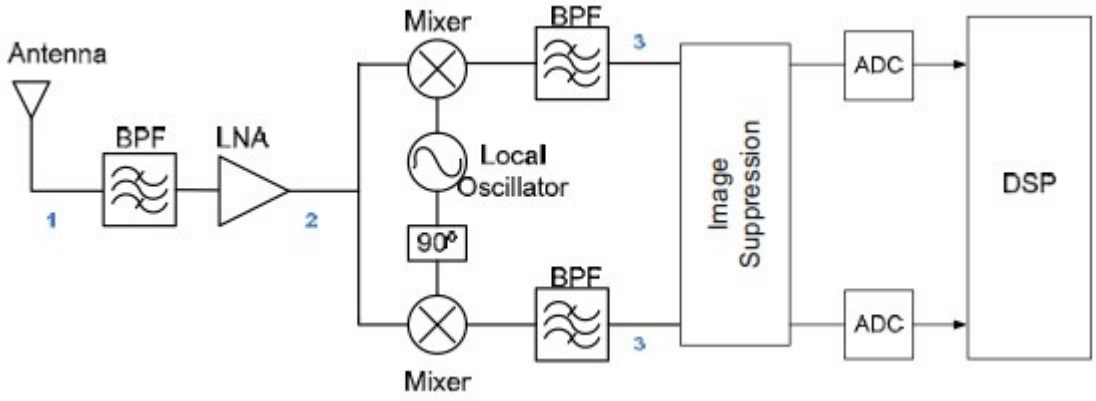


Figure 19: Low IF Architecture [16]

So as we can see from figure 19 what differentiates this architecture from the others is that the conversion to digital domain happens in the IF stage (kHz to MHz) instead of going directly to baseband.

As seen in the homodyne configuration the signal goes through a selection filter in the RF stage which is then amplified by an LNA, after that it's then down converted to Low IF instead of zero IF, and an image suppression block is used to cancel image frequency effects, finally it goes to the ADC for conversion. Some variations of this architecture also place the image suppression block in the DSP.

Advantages

As mentioned, this architecture tries to gather the best of the both previous ones, so it allows a high level of integration while not suffering from DC offset problems at the same time, due to the IF characteristic.

Disadvantages

Although it gathers some of the best qualities, this architecture is not perfect, since it continues to suffer from image frequency problems and I/Q mismatch problems with aggravated effect. Also, a high rate of conversion is needed, therefore the ADC will consume a lot more power.

Of all the architectures that are going to be presented here, this one is probably the most suited for SDR applications.

Band Pass Sampling Receiver

Now, it's presented a different kind of architecture, which is an alternative to all the previous ones. Figure 20 presents the configuration for this architecture.

The first part is fairly equal to all the ones already presented, what changes is that the signal is already converted to digital domain after the LNA through a high sampling rate ADC. So while all the other architectures use analogue components to perform the I/Q demodulation, this one does that section in the digital domain.

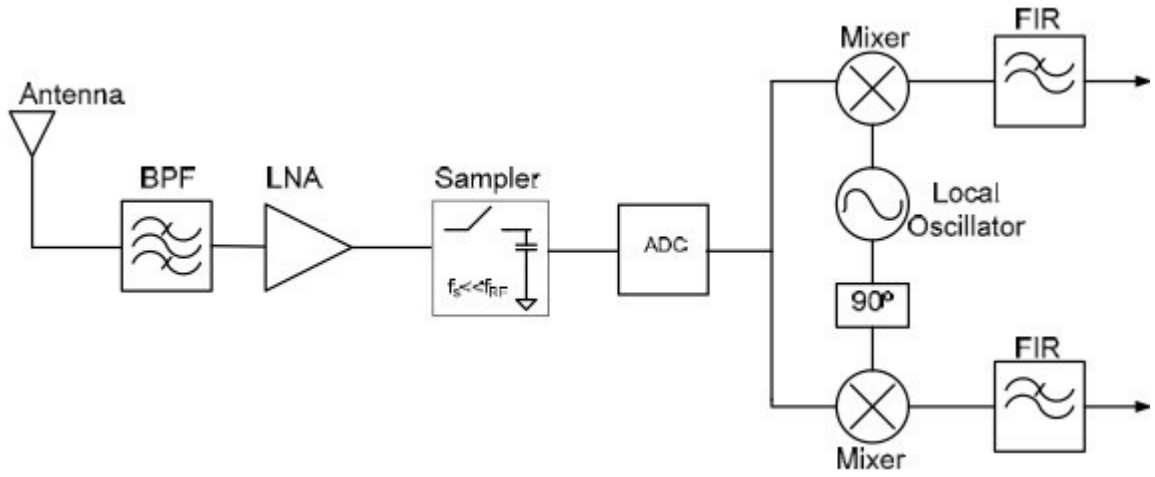


Figure 20: Band Pass Sampling Receiver [16]

Advantages

This configuration presents, right away, many notorious benefits from the last ones. Firstly, I/Q mismatching is reduced, since this function is performed in the digital domain, it's a matter of software tweaking to obtain sufficient matching accuracy in two digital signal paths.

Similarly to the Low IF configuration the signal processing can cut some issues of the analog front end. Another noticeable difference that brings its upsides, is the fact that the mixer is replaced by a sampling circuit, which erases the need of down conversion.

At last there's also a reduction on the number of components needed, since the sampling frequency and processing rate needed are proportional to the information bandwidth, rather than to the carrier frequency.

Disadvantages

All of this can't be accomplished without major crucial requirements. High frequency implementations can generate high clock jitter introducing errors. Also the sample and hold circuit in the ADC must include the RF carrier, which can be of very high demand when regarding the current day's ADC sampling rate.

Other feasible architectures are currently being developed, but these four are considered to be the most important regarding SDR architectures, and between them the Low IF continues to be the most feasible one in terms of SDR based technologies.

This concludes the analysis to the most usual, or well known architectures for front end architectures, and to the chapter as well.

The next chapter will approach the communication end and IoT concepts.

Chapter 3

IoT and Sensor Networks

Since the ultimate goal of the work developed in this master's thesis is to incorporate a spectrum analyzer in a wireless sensor network, for the Internet of Things and the concept of Smart City, this chapter will care to describe some of these concepts a little bit deeper and also analyze some of the most appropriate wireless communication protocols for these kind of applications.

3.1 Internet of Things

This concept is gaining ground in the scenario of modern wireless telecommunications, but what's it about? The Internet of Things (also known as the Internet of Objects) refers to the interconnection of everyday objects.

This connection of equipment and devices to the Internet, makes it possible to monitor and access data from a remote sensor and control it from a distance. That being said IoT is more than a concept, it's a vision for the future. [17] [18] [19].

This vision leads us to more concepts such as **Smart Objects** or even **Smart Cities**. A Smart Object can be defined as everyday objects enhanced by electronic devices that can provide local intelligence and connectivity to the Internet. These electronic devices are computational components attached to the physical object that bridge the gap between the physical and the information world, it is thus described as a cyber-physical system or an embedded system. Some examples are, as seen in figure 21, nowadays objects such as bracelets, thermostats and light bulbs.

Figure 22 also shows an IoT survey on 2015 by the Amsterdam Internal Exchange, and they add to the fact that the use of Smart Objects and attitude towards the IoT is growing from year to year.

Figure 21: Smart Objects [20]

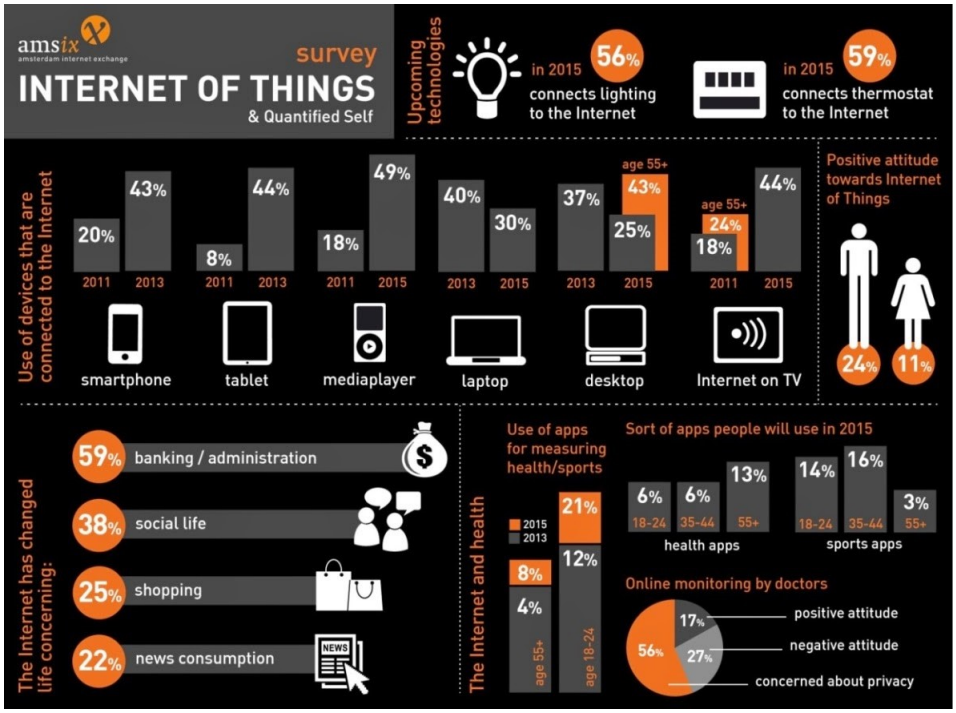


Figure 22: IoT Survey [21]

As for the concept of Smart City, it can have many definitions or even not a certain definition at all depending on different cultures. Nonetheless, when regarding technology and IoT, a Smart City is essentially a city equipped with interconnected Smart Objects, in other words, a city becomes smart or intelligent, when it is instrumented, interconnected, adaptive, autonomous, learning, self-repairing and robust, thus making use of ICTs.

3.2 IoT Scenarios

The immense variety of insurgent smart objects, and consequent smart cities led to many scenarios of applications for the IoT. There are plenty of areas where intercommunication between objects and sharing information can be very convenient, such as:

- **Healthcare:** Measuring patients' or even regular citizens' vital signs through wearable devices such as watches, bracelets and smartphones, allowing emergency units to reach them sooner and more effectively;
- **Tracking:** This feature can be used either for people, animals, vehicles or even personal objects. This can help locate lost people, pets or personal items because the information would be online and accessible to certain people;
- **City monitoring and management:** This allows for the cities' overall monitoring such as electricity grids, water and waste management, as well as highways and traffic occurrences, and one the biggest problems which is pollution;
- **Smart Home:** The number of smart homes is already quite significant nowadays, within this application, a user could easily monitor his house through a smartphone or the Internet, making it possible to switch off devices such as lamps or other household appliances;
- **Smart Agriculture:** Monitor crops in terms of atmospheric conditions and program irrigation systems to reduce amount of work and costs;
- **Spectrum Monitoring:** This application is performed precisely by probes such as the one developed in this project, enabling the transmission in unoccupied bands and better management of the occupied ones.

Applications are actually endless, these are just some of the most common, figure 23 shows how they can interconnect and create their own smart infrastructures within a Smart City.

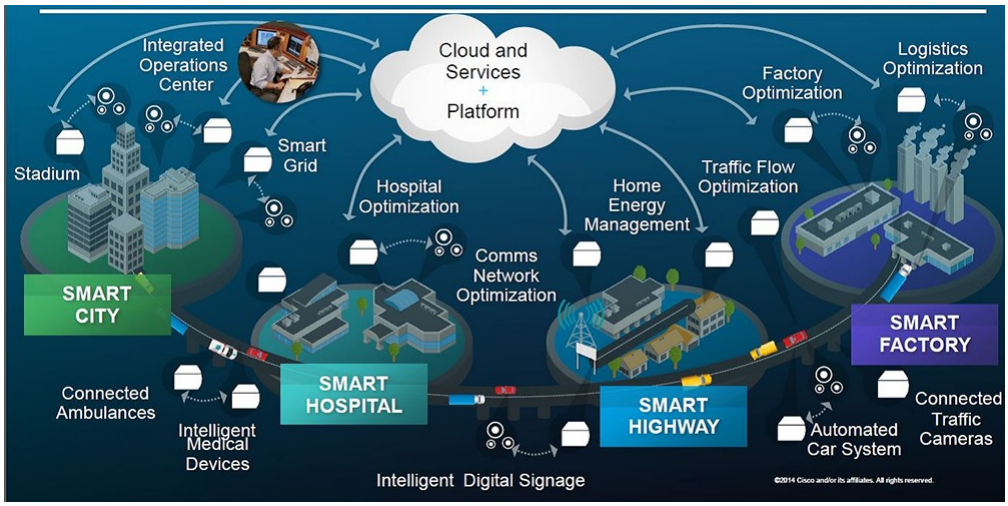


Figure 23: Smart City Applications [22]

3.3 Long-Range Communication Protocols

Even though the IoT has several low and middle range communication standards, such as RFID, NFC, BLE, the tendency is to aim for Long Range communications to better cover the needs inside a city without the need of a close reception point and they also lack the network organization a long range protocol can offer.

In this subsection some of these LPWANs will be approached, namely SIGFOX, Ingenu, and mostly the LoRa protocol which is the one gaining an ever growing momentum currently.

3.3.1 SIGFOX

According to [23] and [24], SIGFOX is the pioneer in LTNs and LPWANs and it's one the main tools of ETSI's efforts to create a standard. It was founded in 2009 by Christophe Fourtet and Ludovic Le Moan.

It uses a variation of the cellular network used in mobile phones, with UNB, BPSK modulated transmissions in which signals have a bandwidth of 100Hz. Because of the narrow bandwidth the noise contribution is very low, thus having high signal power sensitivity (around -142dBm).

Since the layer protocols are kept secret, there isn't much documentation available about its specifications, however some of the first implementations of this protocol claim to have achieved connection with up to a million devices per gateway and a range of 30 to 50 km in rural areas and 3 to 10 km in urban areas. The ultimate goal for SIGFOX is to become a global IoT operator.

3.3.2 Ingenu

This protocol was founded in 2008 with the name On-Ramp Wireless, which was changed to Ingenu in September 2015 and it is one of the youngest technologies of the sort. The protocol adopts a star networking topology with access points acting as coordinators of end points and the system supports both uplink and downlink transmissions, although favoring the uplink. [23], [24], [25]

The company developed and patented a new multiple access scheme for communication with base stations called RPMA, a variation from the conventional CDMA, that serves as a point of differentiation from the rest of the competitors. [25] RPMA differs from CDMA in two aspects: All the nodes use the same Gold code to spread the bits before transmitting and also the transmissions are asynchronous and start with a random delay, an illustration of this transmission process can be seen in figure 24.

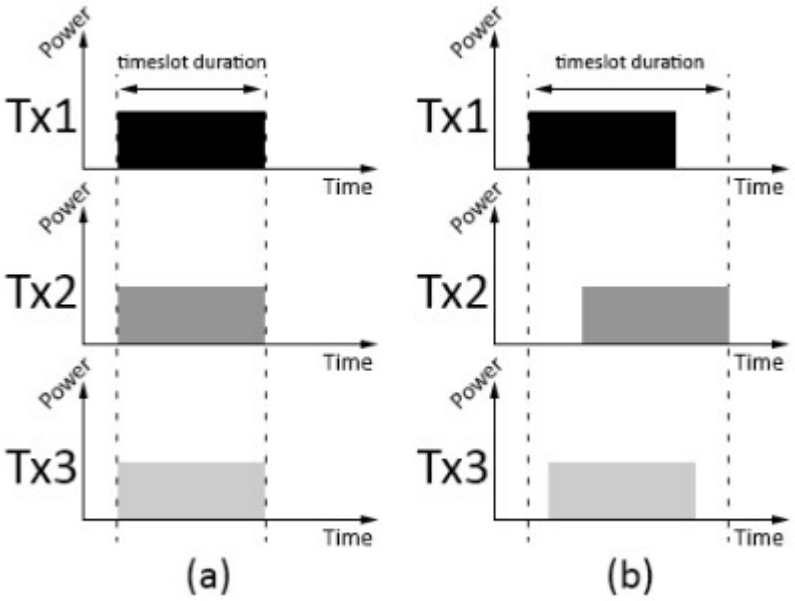


Figure 24: CDMA (a) and RPMA (b) Transmission process comparison [23]

Ingenu operates in the 2.4GHz band contrarily to the other LPWAN technologies, and it's able to work over long range links and under tough RF surroundings.

3.3.3 LoRa and LoRaWAN

LoRa stands for Long Range and LoRaWAN is an LPWAN specification intended for wireless battery operated devices directed for IoT applications.

However, an aspect should be clear from the start: LoRa and LoRaWAN should not be confused, because they are different concepts.

- **LoRa** refers only to the link layer protocol, and it's suited for P2P communications between nodes. It can also refer to the modulation technique implemented in LoRaWAN networks, proprietary from Semtech Corporation which also manufactures the chipsets that run this protocol. There are modules that only operate with this P2P LoRa protocol, usually they're cheaper than the LoRaWAN ones;

- **LoRaWAN** includes the network layer too, so with these kind of modules it's possible to send information to any Base Station and implement a sensor network, needless to say, they are more expensive than exclusive LoRa modules. LoRaWAN can operate in the 433, 868 or 915MHz ISM bands depending on the region it is employed, for example in Europe, it employs either GFSK or the previous proprietary LoRa modulation protocol. It can also implement FSK, MSK, GMSK and OOK modulation. [26]

Now it's important to enumerate some of main specification of the LoRa system as a whole technology. [23] [24]

- Modulation employs a version of CSS
- Bandwidth of 125kHz
- Payload ranging from 2 to 255 bytes
- Data rates ranging from 0.3Kbps to 50Kbps
- Low cost and low power
- High sensivity: down to -148dBm on official Semtech modules
- Network follows a hierarchical star based architecture (Figure 25)
- Coverage area can go up to 22km in rural sites and up to 2km in urban areas
- While the PHY layer of LoRa network is proprietary, the network protocol (LoRaWAN), is open source, and its development is made by LoRa Alliance, created by Semtech.

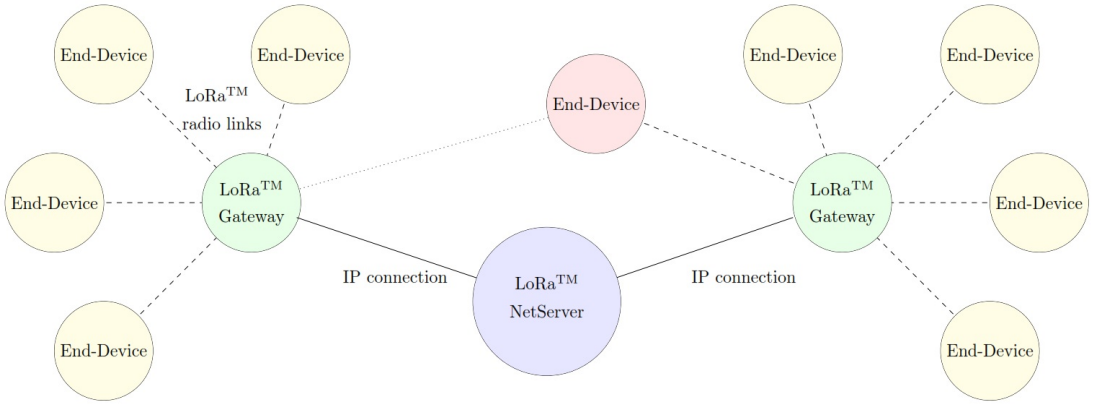


Figure 25: LoRa's Star Based Topoly [24]

From [27] we see that LoRaWAN has three classes of operation: A, B and C:

- **Bi-directional end-devices (Class A):** The transmission slot scheduled by the end device is based on its own communication with a small variation on a random time basis. It's the lowest power end device operation, and it's suited for applications that only require downlink communication after the end device has sent an uplink transmission.
- **Bi-directional end devices with scheduled receive slots (Class B):** Adding up to Class A, class B devices open extra windows at schedule times allowing the server to know when the end device is listening.
- **Bi-directional end devices with maximal receive slots (Class C):** This has nearly continuous open receive windows, that close only when transmitting.

3.3.4 LTE-M

Although not an LPWAN, LTE-M is definitely an honorable mention. We're certainly aware that LTE is already an established protocol for mobile communications, the only thing necessary would be to upgrade the existing infrastructures. LTE supports FDD and TDD modes using a common sub frame structure of 1ms and such a short length allows for low latency ensuring a good user experience.

Very shortly, LTE-M is an evolution of LTE optimized for IoT in 3GPP RAN. It was first released in late 2014 and further optimization was included in early 2016. 3GPP has specified low cost M2M devices, known as Cat-0. This standardization is gaining momentum, enhancing coverage, battery life, and lower complexity compared to the current LTE devices. [27]

3.3.5 Section summary

SIGFOX intends to become a global IoT operator. Although a high risk approach, it's low cost, available today, but research has still much to evolve to prove its success.

Ingenu has the distinct feature of having developed their own multiple access protocol (RPMA), although the verdict on whether it out performances CDMA is arguable [23], it can definitely have its benefits.

LoRa stands for a more distributed concept, since they're letting mobile operators roll out infrastructures as well as allowing it to be used for public and private networks, and it has the advantage of its specifications being more well-known than SIGFOX for example. It also has few financial risk, since it has both public and private network ends, even if the first fails the second will not, therefore they'll continue selling and this protocol will be an option during decades.

LTE -M has the advantage of only needing an upgrade form the conventional LTE, but that feature can also be a problem since as it is a subset of LTE, it still need to carry a lot of additional silicon and protocols to allow it to coexist with other LTE users. Also LTE modems are as expensive as they're complex and kind of hard to maintain. [28]

3.3.6 Performance comparison between LPWANs

Focusing more on coverage and data rate comparisons, figure 26 relates the study performed in [24] in which LPWAN radio technologies are put to the test revealing some interesting results, implying that some specifications can outperform the ones theoretically stated on the developers' datasheets and sites.

	SIGFOX™	Ingenu™	LoRa™
Coverage range (km)	rural: 30–50 urban: 3–10	≈ 15	rural: 10–15 urban: 3–5
Frequency bands (MHz)	868 or 902	2400	various, sub-GHz
ISM band	✓	✓	✓
Bi-directional link	✓	✗	✓
Data rate (Kbps)	0.1	0.01–8	0.3–37.5
Nodes per BS	$\approx 10^6$	$\approx 10^4$	$\approx 10^4$

Figure 26: LPWAN performance comparison [24]

3.3.7 Tradeoff comparison: LPWAN vs Short Range

Some key aspects while considering network connectivity include range, data rate, power, frequency and security. This section started off by implying LPWANs are more valuable and useful for IoT than short range communications. The following table shows a select number of trade offs one has to consider while choosing the kind of protocol you want to apply to the desired network

Technical capabilities	Low Power Wide Area Networks (LPWAN)							Short Range Networks				
	LoRaWAN	Neul	NWave	SigFox	Weightless -N	Weightless -P	Cellular	BLE	WiFi	Thread	ZigBee	Z-Wave
Range (km/m)	2-5 urban; 15 suburban; 45km rural	up to 10km	up to 10km	up to 10km urban; 50km rural	5km	2km	35km GSM; 200km 3G/4G	80m	50m	Mesh	100m/Mesh	30m/Mesh
Deep Indoor Performance	Yes	ISM yes, Whitespace no	Yes	Yes	Yes	Yes	No	No	No	No	-	-
Freq. Band	Varies. Sub-GHz	ISM or Whitespace	Sub-GHz	Frequency independent; 868/902MHz	Sub-GHz	Sub-GHz	900/1800/1900/ 2100MHz	2.4GHz	2.4GHz	2.4GHz	915MHz/2.4GHz	900MHz
ISM?	Yes	Yes, depends on base-station	Yes	Yes	Yes	Yes	Depends	Yes	Yes	Yes	Yes	Yes
Fully Bi-Directional	Yes, depends on mode	Yes	No	No	Uplink only	Yes	Yes	Yes	Yes	-	Yes	Yes
Data Rate	0.3 - 50 kbps adaptive	10 - 100kbps	100bps	10 - 1000bps	30kbps - 100kbps	up to 100kbps adaptive	35-170kbps GSM/ 3 - 10mbps LTE	< 1mbps	600mbps max	-	250kbps	10 - 100kbps
Power Profile	Low	Low	Low	Low	Low	Low	Medium	High	High	Low	Low	Low
Authentication	Yes	-	Yes	Yes	Yes	Yes	High security, back by major telecoms	Trusted devices problematic	Yes	Yes	Yes	Yes
E2E Encryption	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Over the Air Software Upgrades	Yes	-	No	No	No	Yes	Yes	Yes	Yes	-	Yes	Yes
Supports sensors moving between hubs	Yes	-	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes, mesh-based	Yes, mesh-based
Location Aware	Yes	-	No	No	No	-	Yes	No	Yes	-	-	-
Operational Model	Public or private	-	Public or private	Public	Public or private	Public or private	Public or private	Public or private	Public or private	Private/WiFi backbone	Public or private	Public or private
Standard	LoRaWAN	Weightless	Weightless	No	Weightless	Weightless	GSM, LTE etc	Bluetooth 4.0	IEEE802.11	Thread, based on 6LoWPAN IEEE802.15.4	ZigBee	Z-Wave
Scalability	Yes	Yes	Yes	Yes	Limited	Yes	Yes	Yes		Yes	Yes	Limited

Figure 27: Tradeoff comparison between LPWANs and Short Range Communications [29]

3.3.8 Known Issues

Even though the enabling of all the technologies referred in the above sections, and the IoT implementation being feasible, there is still a lot of research to be done in this field. The image below, as shown in a survey of the IoT, states the greatest issues that still exist with factors like standardization activities, IoT scenario requirements and networking issues.

Open issue	Brief description of the cause
Standards	There are several standardization efforts but they are not integrated in a comprehensive framework
Mobility support	There are several proposals for object addressing but none for mobility support in the IoT scenario, where scalability and adaptability to heterogeneous technologies represent crucial problems
Naming	Object Name Servers (ONS) are needed to map a reference to a description of a specific object and the related identifier, and *vice versa*
Transport protocol	Existing transport protocols fail in the IoT scenarios since their connection setup and congestion control mechanisms may be useless; furthermore, they require excessive buffering to be implemented in objects
Traffic characterization and QoS support	The IoT will generate data traffic with patterns that are expected to be significantly different from those observed in the current Internet. Accordingly, it will also be necessary to define new QoS requirements and support schemes
Authentication	Authentication is difficult in the IoT as it requires appropriate authentication infrastructures that will not be available in IoT scenarios. Furthermore, things have scarce resources when compared to current communication and computing devices. Also man-in-the-middle attack is a serious problem
Data integrity	This is usually ensured by protecting data with passwords. However, the password lengths supported by IoT technologies are in most cases too short to provide strong levels of protection
Privacy	A lot of private information about a person can be collected without the person being aware. Control on the diffusion of all such information is impossible with current techniques
Digital forgetting	All the information collected about a person by the IoT may be retained indefinitely as the cost of storage decreases. Also data mining techniques can be used to easily retrieve any information even after several years

Figure 28: Known Issues of IoT communication technologies [30]

Chapter 4

System Project and Implementation

4.1 Requirements and Architecture

The goal of the project in this thesis is to develop a portable spectrum probe, capable of performing spectrum sweeps in the biggest frequency interval as possible, and send the sensor information to a computer terminal.

For these end goals, it'll be necessary to acquire an **RF front end**, implemented with one of the architectures mentioned in subsection 2.3.5. The preferred architecture for the front end receiver would be a **Low-IF configuration**. It's essential to consider some specs when choosing this module, namely the dynamic range and frequency range, the receiver architecture implemented and power consumption. This block should output the power measured for the spectrum at a certain frequency. Depending on the chosen chip it will be programmed through a serial protocol such as SPI or I^2C, so the MCU has to support it.

The **wireless communication** block requires, at least, a receiver and a transmitter pairing modules that employ a low power and low cost protocol with a long range of communication and high coverage area, although transceivers would allow for remote radio reconfiguration and parameter choice, nonetheless for the purpose of this thesis, a receiver / transmitter pair would be sufficient. Regardless of this choice, the module should be low power, low cost, and employ a long range wireless communication protocol. One of the modules should connect to the MCU via serial interface and receive the data it must then send to the terminal.

Since this is supposed to be a portable sensor device, one must take in consideration that the system should probably be powered between 3.3 V and 5 V, considering that those are the standard values used by most modules and battery supplies.

Both the RF front end and the communication modules should be connected to a central unit (an MCU). This unit should be capable of programming / reprogramming the RF front end chip and also read the output of the system, so that it can send these data to the communication modules. The processing of the data should be done in the computer terminal for efficiency purposes.

The **user end** should contain at least a start and stop button, for the samples to be collected from the sensor and displayed in a real time plot. A graphic with the results would also help a better understanding of the spectrum at a given frequency range.

4.1.1 Proposed Architecture

The proposed solution to better fulfill the project requirements is composed by an RF front end, responsible for capturing the RF spectrum, which will be programmed by a serial protocol implemented in the MCU. The RF front end should also send the collected data to the MCU which will simultaneously dispatch these samples to the Wireless Module, responsible for then sending the samples to the receiver. The Wireless Receiver will then proceed to send the samples, through a serial protocol to the PC terminal where they will be processed and turned into the desired interface.

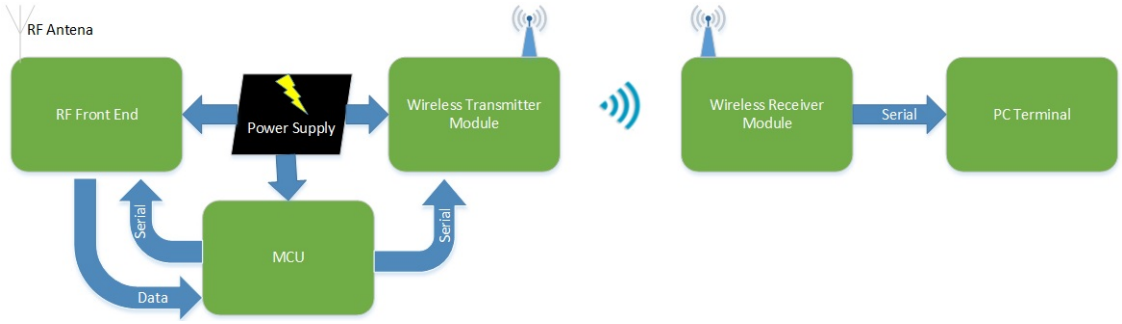


Figure 29: System Architecture

4.2 IoT modules

After some research one of the most logic choices would be one of the SX127x series Low Power Long Range Transceiver from Semtech that employs the LoRa protocol. The key product features are as follows:

- LoRa Modem
- 168 dB maximum link budget
- +20 dBm – 100 mW constant RF output
- +14 dBm high efficiency PA
- Programmable bit rate up to 300 kbps
- High sensitivity: down to -148 dBm
- Bullet proof front end: IIP3 = -11 dBm
- Excellent blocking immunity
- Low RX current of 9.9 mA, 200 nA register retention
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK, LoRa
- Built-in bit synchronizer for clock recovery

- Preamble detection
- 127 dB Dynamic Range RSSI and OOK modulation
- Automatic RF Sense and CAD with ultra-fast AFC
- Packet engine up to 256 bytes with CRC
- Built-in temperature sensor and low battery indicator

Below there's a block diagram on this module's architecture:

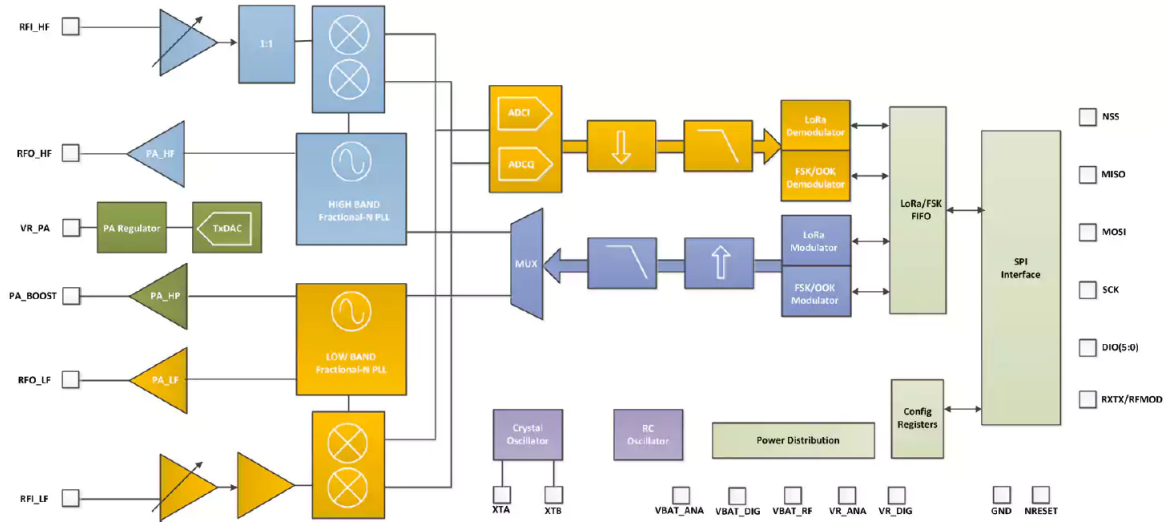


Figure 30: SX127X Series Block Diagram [31]

However, the development kits for these modules are quite expensive (around 50€), not to mention the specs on this Transceiver are a bit overkill for the purpose at hand.

After some more research, new modules have been discovered, and they were already in stock in the Institute of Telecommunications headquarters which would speed up the process of testing. These modules are the DRF4432S and DRF5150S. They are a pair of receiver / transmitter modules manufactured by Dorji applied Technologies. The modules are illustrated in figure 31.

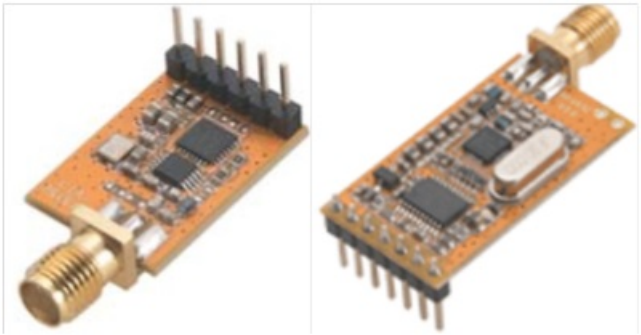


Figure 31: Dorji DRF5150S and DRF4432S modules [32] [33]

These are some of the specs featured on these modules:

- GFSK receiver and transmitter modules
- ISM frequency band
- 81 Kbps data rate
- Multiple channels
- -120 dBm sensitivity in the receiver
- 10 dBm output power for the transmitter
- Configurable Baudrate
- 256 bytes data buffer
- Standby current less than 3 μA for the receiver and 2.5 μA for the transmitter
- Supply voltage from 3.4 to 5.5 V to the receiver and 2.1 to 3.6 V for the transmitter

These specifications make them ideal for sensor type applications and they're capable of being implemented in a wireless sensor network, which is our goal. Even though they're not certified LoRa protocol modules, they employ an equivalent one, based in GFSK, transmitting in the unlicensed ISM and LPRD bands. They're also very low cost which is a big plus side.

DRF4432S is used together with the DRF5150S transmitter module, the first collects data from the second. The transfer rates and all other configurations are made by a software called DRF Tool 5150 that uses UART interface. The device that allows this UART interface is the DAC-02 shown in figure 32. For more information on these configurations, software and operation modes, please check Appendix B.1.

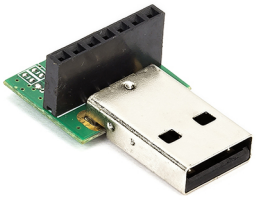


Figure 32: TTL -USB Converter Board DAC 02-5

4.3 RF Front End

For the choice of the front end, the first aspect to consider was the desired RF receiver architecture to employ, from section 2.3.5 we can conclude that the most suitable architecture for an SDR based project is the Low IF receiver, and this was actually the first choice for the architecture. However, at this stage, to prevent any more delays, it was decided to use an already assembled board by researcher and co advisor Pedro Cruz. The front end, ended up having a Homodyne configuration, and the choice for that matter was MAXIM's MAX2112 Direct Conversion Tuner for DVB-S2 Applications. This chip is a low-cost, direct conversion tuner IC designed for satellite set-top and VSAT applications. Some of the main features of this specs are listed below:

- 925 MHz to 2175 MHz Frequency Range
- Monolithic VCO with Low Phase Noise: -97 dBc/Hz at 10 kHz and No Calibration Required
- High Dynamic Range: -75 dBm to 0 dBm
- Integrated Variable BW LP Filters: 4 MHz to 40 MHz
- Single +3.3 V ±5% Supply
- Low-Power Standby Mode
- Address Pin for Multituner Applications
- Differential I/Q Interface
- I^2C 2-Wire Serial Interface
- Very Small 28-Pin TQFN Package

The functional diagram of this chip is presented in figure 33

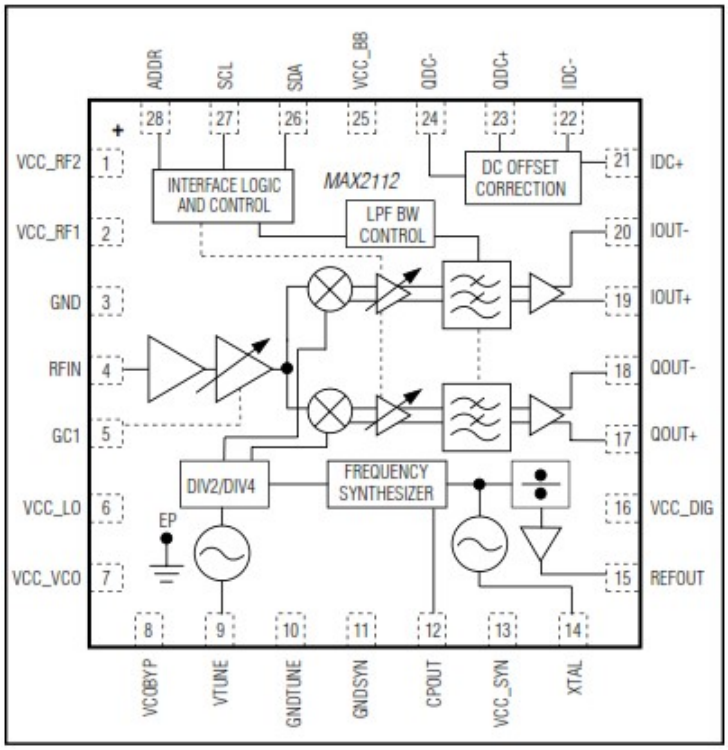


Figure 33: MAX2112 Functional Diagram [34]

As can be seen the chip includes an LNA, and an RF variable gain amplifier, I and Q down converting mixers, and baseband LPFs with programmable cutoff frequency control and digitally controlled baseband variable-gain amplifiers, all together the RF and baseband variable-gain amplifiers can provide up to 75 dB of gain control range.

The MAX2112 includes fully monolithic VCO and a complete Fractional-N frequency synthesizer. The IC includes a VCO with VAS function, automatically selecting the proper VCO. For a more appropriate explanation on VCO and frequency synthesizers check the next subsection. Although the frequency range doesn't even cover half the spectrum portion intended in the beginning it's still very good for demo purposes. The featured frequencies comprise some important communication technologies like GSM, GPS, AM radio, Radars (RVA, RLC), IMT communications, among others.

This chip features a set of 12 programmable registers to deal with the frequency synthesizers, VCO, PLL and also the configurable baseband gains and LPF. This process is explained thoroughly in the Appendix A.1.

There is a special need of this chip which is to have a controllable voltage (VGC1) that's responsible for setting the initial variable gain attenuator.

The followed layout for the application circuit assembly was the one presented in figure 34.

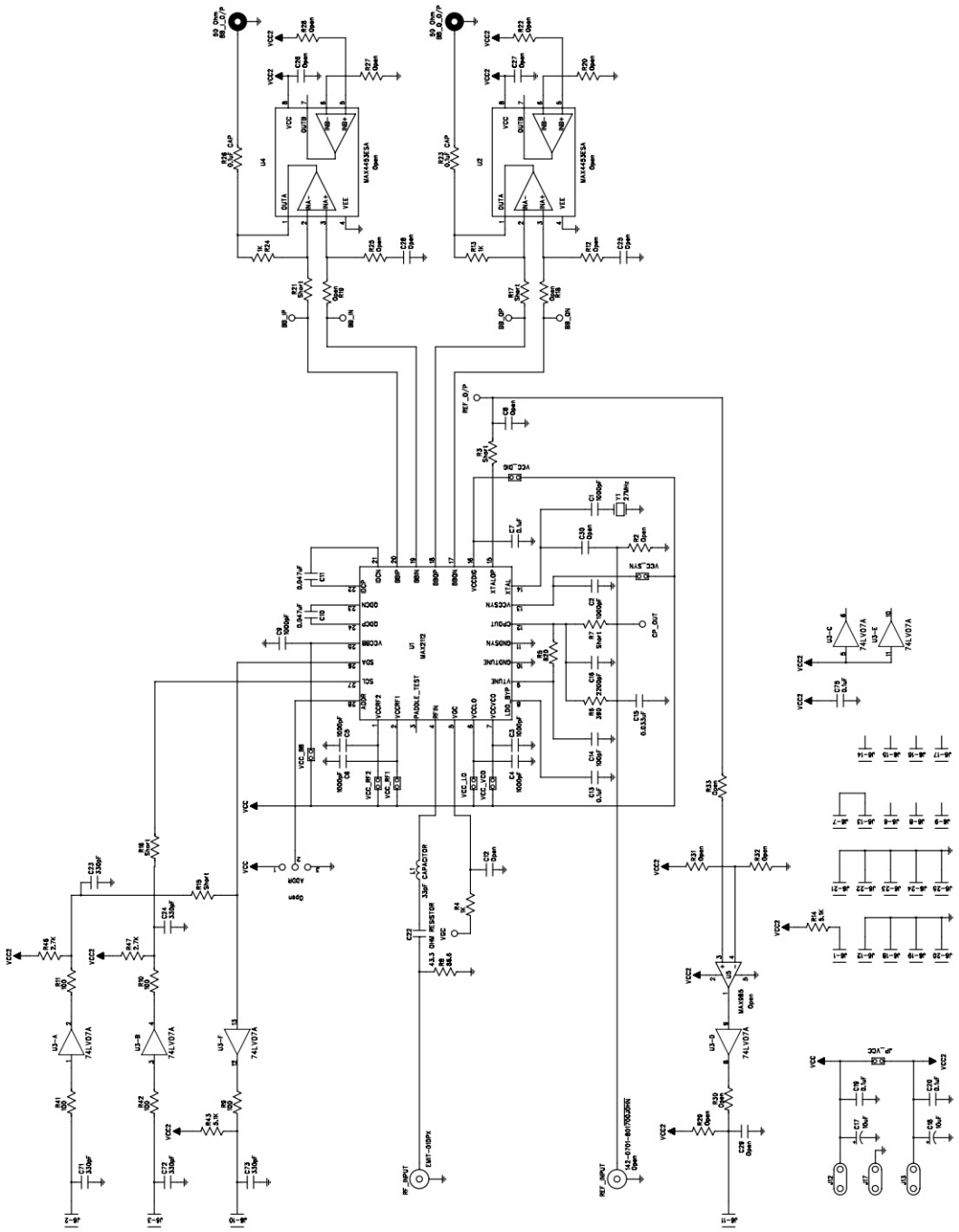


Figure 34: MAX2112 Board Schematic Layout from [35]

The only major change to this schematic was the removal of the two buffers in the end that were saturating the output signal. This is the recommended application circuit referred in the Evaluation Kit datasheet [35].

The end result of this RF front end is shown on the photo below:

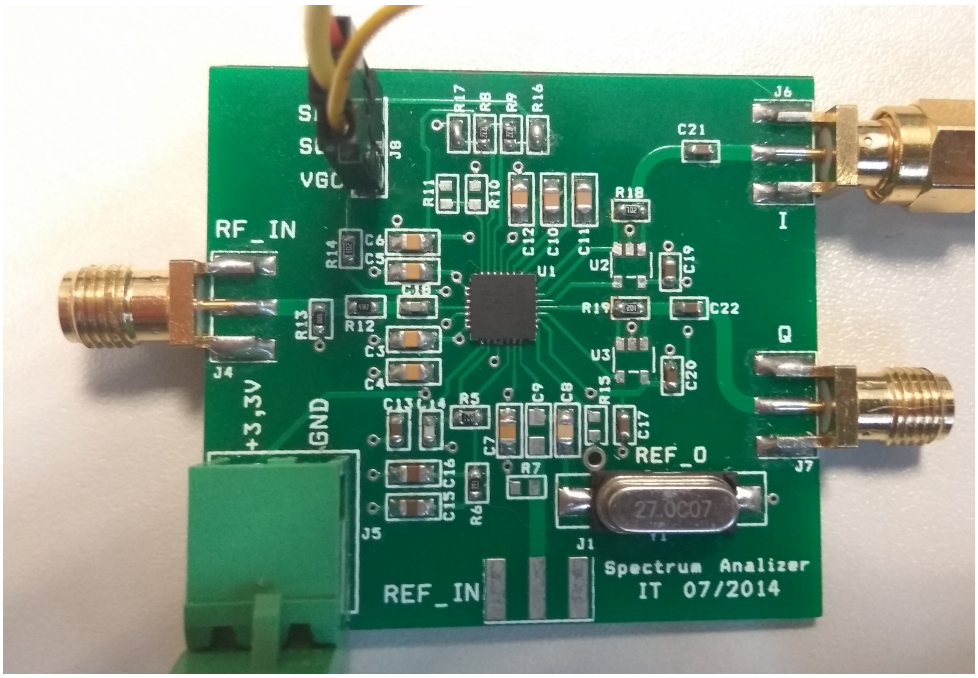


Figure 35: RF Front End with MAX2112 chip

Again, it's worth mentioning that this PCB was designed by co advisor Pedro Cruz, a work that preceded the beginning of this dissertation.

4.3.1 PLL's

The most important part to understand about the functioning of the MAX2112 chip is the PLL block, as they play a vital role in many of modern day circuits such as this one. Essentially they are used to demodulate amplitude and frequency modulated signals, synchronizing clocks, recovering signals from noise, and other applications.

The basic concept is shown in the diagram below, and it's composed by 4 main blocks: the phase detector, the VCO, the Loop filter and the frequency divider.

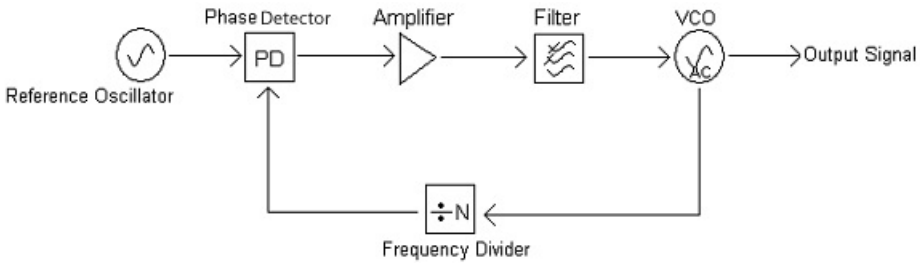


Figure 36: Phase locked loop basic diagram [36]

Phase Detector

The reference signal and the signal coming from the VCO are both connected to the phase detector. This block is responsible for calculating the phase difference between the reference signal and the signal from the VCO, this will generate an error voltage proportional to that difference.

Loop Filter

Before entering the VCO, the error voltage is pre amplified and then filtered in this Loop Filter, eliminating high frequencies and transforming it into a DC voltage used in the VCO.

VCO

The filtered voltage proceeds to the voltage controlled oscillator. As the name states this oscillator is tuned by a voltage, which is the error voltage from the filter. The output of the VCO will be a compensation frequency, proportional to the error voltage, that will reduce the phase difference between the two signals. [37] The name Phase Locked Loop comes from this stage, because the loop won't be locked until the VCO can no longer reduce the phase error between the two signals.

The number of available VCOs varies from device to device, the MAX2112 provides 24 VCO with autoselection, that selects the more suited VCO for a given frequency.

Frequency Divider

This block divides down the frequency into integer number parts. The function of this block is, so to say, to "fool" the phase detector into "thinking" the VCO was operating on a lower frequency, which will provide flexibility to operate the VCO at an actual arbitrary frequency, otherwise we wouldn't need anything but a crystal oscillator as reference.

A clearer example is to suppose we want to operate at a frequency of 40MHz, but the crystal oscillator is only 10MHz. By dividing the frequency by a factor of 4 we can make the phase detector think the VCO was only operating at 10MHz when it was really four times that amount, so in the end when the loop is locked the VCO would by oscillating at 40MHz as stable as the crystal reference.

The MAX2112 provides two dividers: the N and F divider, which basically can divide frequency into smaller portions allowing the user to use basically whatever frequency he wants. [38]

4.4 Power Detector

Since the RF front end doesn't include a power detector, this section deals with the design of a peak detector. The proposed schematic is as follows:

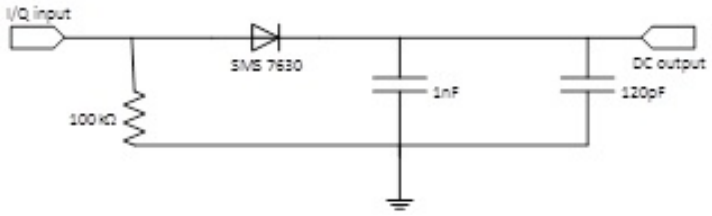


Figure 37: Peak Detector

The SMS 7630 is a surface mount Schottky Diode which is a zero-bias diode allowing for low voltages to conduct the diode, therefore enhancing the DC output dynamic range. The capacitors are there for filtering purposes and the resistor is to avoid feedback effects (without it there'd be no signal in the output).

4.5 Low Pass Filter

Given the special need of the MAX2112 to have a voltage that controls the attenuation, the most attractive option is to have that voltage being automatically controlled by software. To do so the MCU will need to generate a PWM signal that must be posteriorly rectified by an LPF in order to turn that voltage into DC.

For this effect a simple RC filter with a cutoff frequency near to 1 Hz and a settling time around 200 ms will suffice, the proposed configuration was the following:

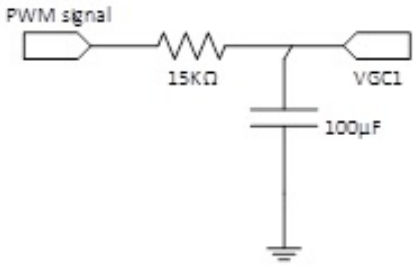


Figure 38: RC Low Pass Filter

4.6 MCU

Considering the kind of configuration, the MAX2112 uses, and to run the entire RF circuit as a spectrum analyzer, automatic functions are very appealing to process and control electrical parameters.

There are several microcontroller platforms with extensive features and specifications. Nonetheless in the scope of this thesis it was decided to use the Arduino IDE with an Arduino Uno, since it's an open-source electronics prototyping platform based on feasible and easy-to-use hardware and software. The specifications for the Arduino Uno are presented in the table below.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figure 39: Arduino Specifications

These specifications are more than enough to support the spectrum sensor. It provides I^2C data and clock lines to program the MAX2112 and RX /TX pins for UART interface with the Dorji Modules. The summary of functions performed by the Arduino are:

- I^2C serial programming of the MAX2112 module to perform frequency sweeps
- Write a digital voltage to the MAX2112 that is needed to control the attenuation of the amplifier
- Read the DC output of the LPF and adjust the gain accordingly
- UART interface with the Dorji wireless modules to send the data

4.7 PCB Design

The available Arduino Uno for this project is the R3 board, which is somewhat big and offers the chance to remove the ATMEGA328P chip, it was decided that a PCB board was better suited for this project's application. So basically this PCB design requires designing a "standalone" Arduino Board with just the ATMEGA chip in it.

In figure 40 we can see the basic components and operations for building this standalone circuit, which are: a 16 MHz crystal for the oscillator that will connect to pins 9 and 10, along with two 22 pF capacitors connected to GND, a 10k resistor to pull up the reset line, and the respective voltage supplies to GND and VCC pins.

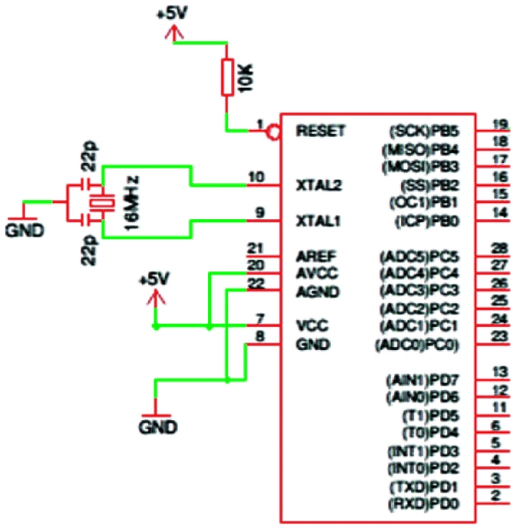


Figure 40: ATMEGA328P Standalone Basic Circuit

On top of this design we must add the remaining blocks that can be designed within the PCB board to the schematic. They are the LPF and header pinouts for the IoT module and remaining connections, since the MAX2112 and Power Detector are already implemented in their own boards we'll just attach them in the final system. All the information about the ATMEGA328P was consulted in its datasheet [39].

For the PCB to be of practical use, a **reset button** should be implemented to restart the program on demand, and since the ATMEGA is supplied at approximately 5 V. We also need a **voltage regulator** to lower this voltage to 3.3 V so the DRF5150S and MAX2112 can be properly fed. The chosen regulator was an LT1761ES5-3.3TRMPBF which is a low dropout regulator suited for battery supplied applications. [40]

As for the **supply**, 3 AA 1.5 V batteries should be enough for this purpose, providing around 4.5 V at maximum charge. To avoid a clumsy on / off switching by removing the batteries, a switch was added at the terminal of the batteries. The end schematic with all the need components is shown on figure 41.

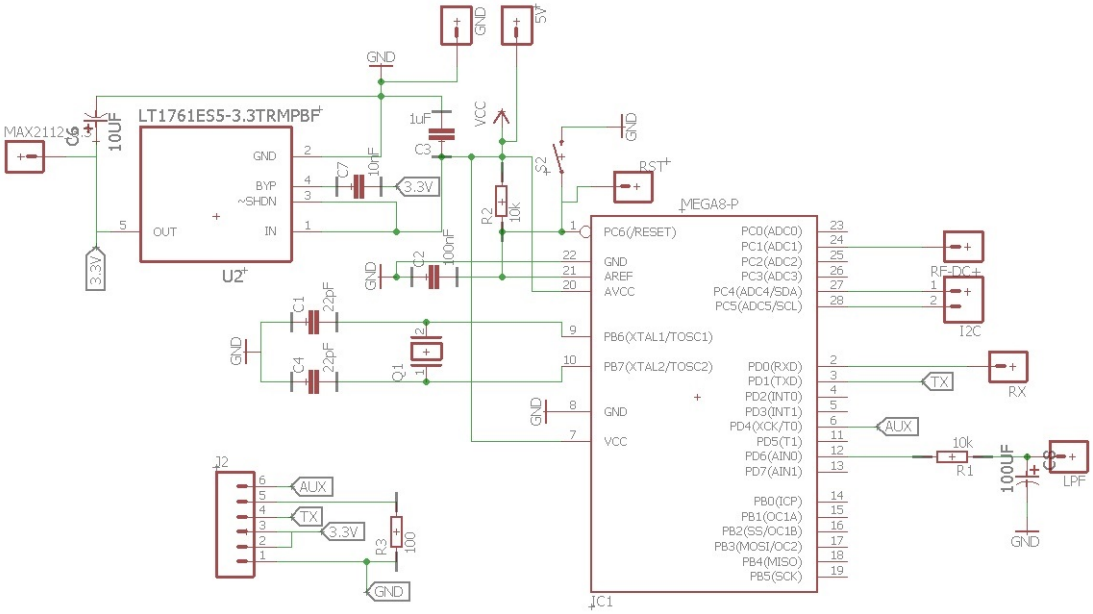


Figure 41: Schematic Design for Final System

The corresponding Board Layout and component arrangement is presented below. All circuitry and layout was designed using Eagle CADSOFT 7.7.

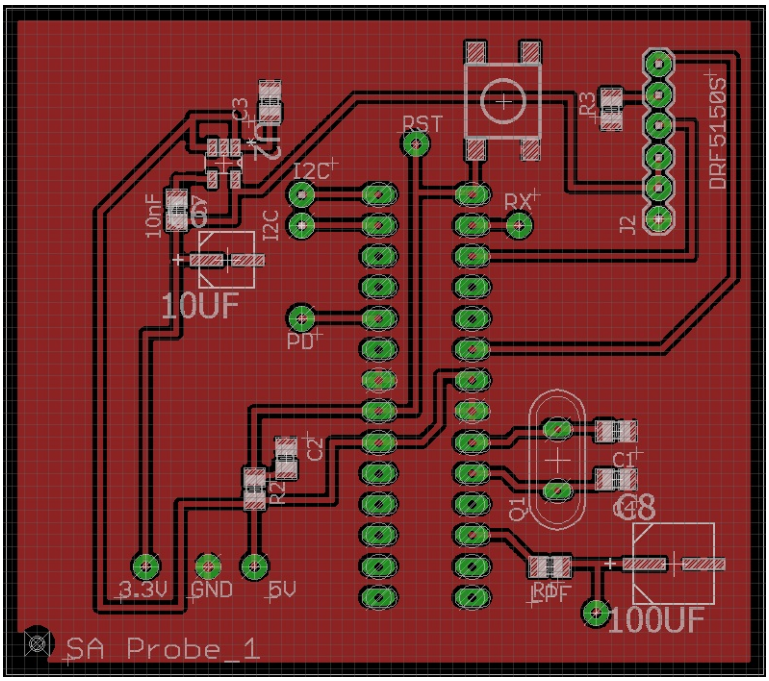


Figure 42: PCB Board Layout for Final System

After printing the PCB looks as shown below in image 43. Note that there are some differences from the schematic and board above, because there was a previous version with some wrong connections, as there wasn't time to print a new one, some patches were done to the old design in order to make it work properly.

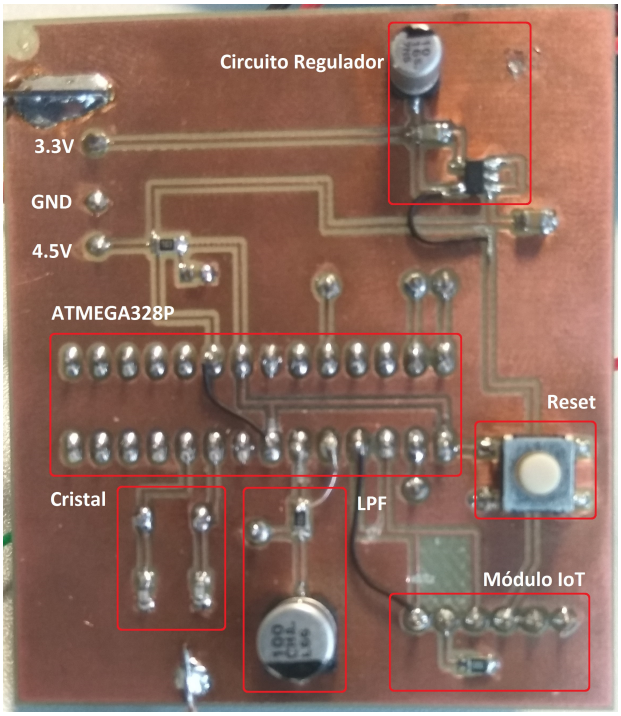


Figure 43: PCB

4.8 Final Spectrum Sensor Electronic Model

With all the blocks designed and implemented the end result was the system in image 44. Figure 45 shows the receiver on the user end with the DRF4432S receiver. A breadboard implementation suffices, even more it simplifies the reprogramming of the module if needed. To reprogram the module all we have to do is remove the module and connect directly to the DAC-02 USB module.

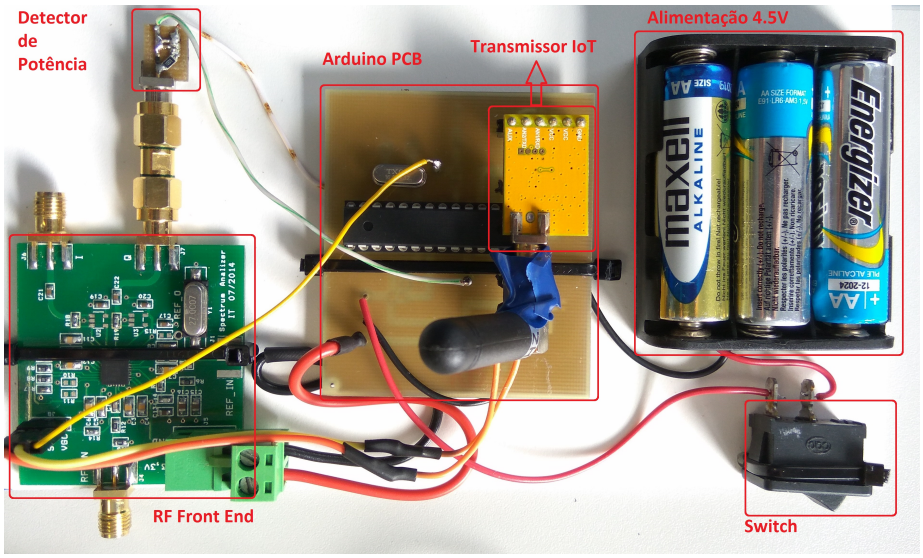


Figure 44: Final System Arrangement

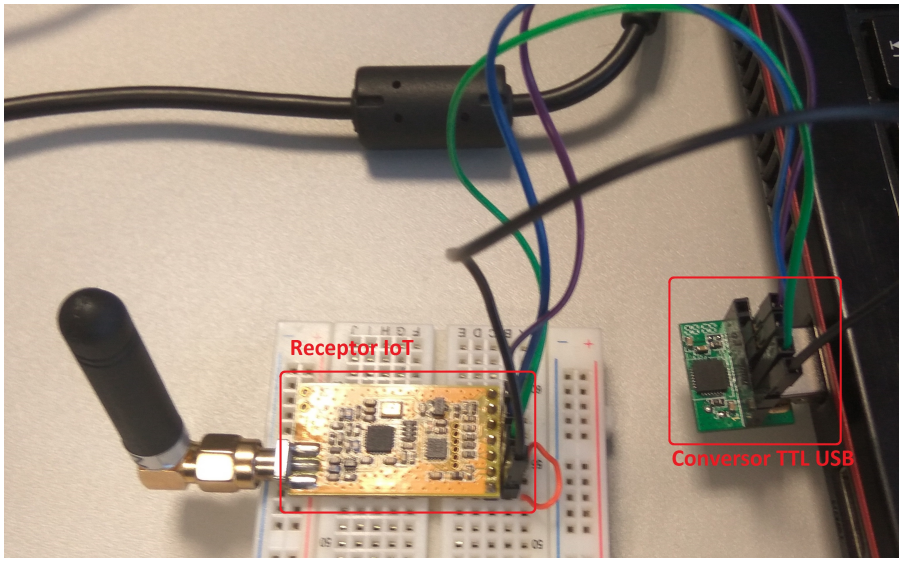


Figure 45: User receiver end with DRF4432S

4.9 Software and Algorithms

This section's purpose is to briefly describe the process of programming each block to serve its purpose in the system.

4.9.1 MAX2112 and Arduino

For the programming of the front end, Arduino was chosen as the most practical platform for the effect. Figure 46 depicts the flow chart of the algorithm performed in the Arduino IDE.

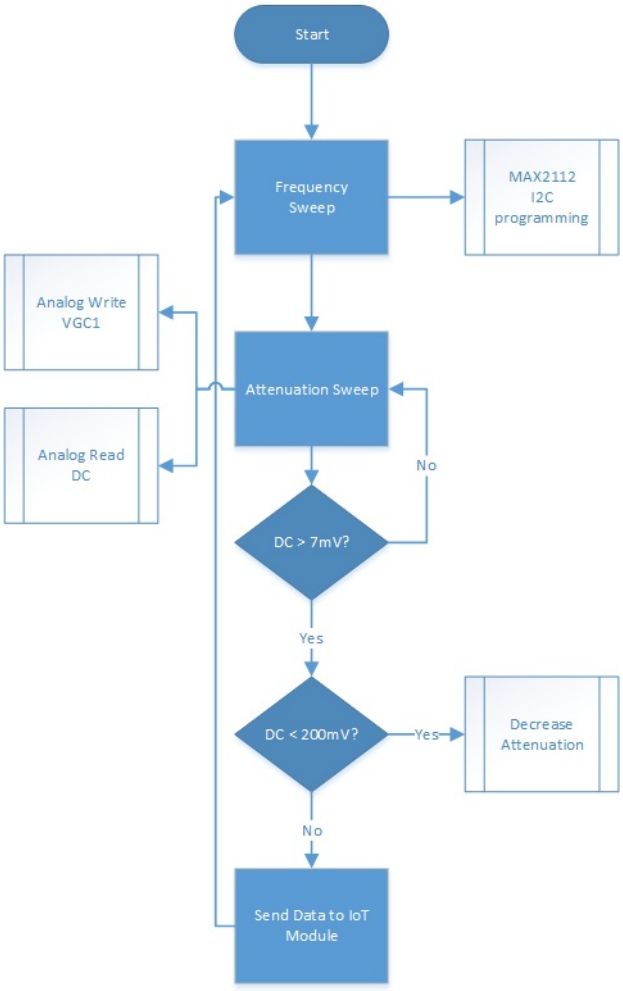


Figure 46: Arduino IDE Flowchart

Basically the program starts by initializing all connections: I^2C for MAX2112, UART Serial Interface with DRF5150S transmitter module, and the GPIO configuration for Analog Read and Write for the LPF output and VGC1 write, respectively.

After all initial configurations are done the main program kicks in and sweeps the programmed frequency range to search for significant power in the spectrum. This is where the MAX2112 chip is programmed via I^2C. For more detailed information on this programming algorithm refer to Appendix A.1.

Once the frequency sweep is initiated, a sweep must also be made with the configurable attenuation voltage. It starts at its maximum value and lowers until it finds significant DC output from the received signal. This process also generates two subsequent ones, which are the Analog Read and Write.

After stabilizing the DC value in a range somewhere around 200 mV the data is now ready to be sent through serial to the transmitter module. A value between 200 mV and 300 mV is preferred because it positions the detector in its most linear zone, this can be verified further ahead in graphic 67 in the results section.

At last the process starts over and it moves on to the next frequency.

4.9.2 IoT modules

The chosen IoT modules did not provide material for manual programming, or in other words, code writing to setup the communication protocol. Instead Dorji provides a software that allows for the programming of both the receiver and transmitter to take place, the modules communicate with the software through an acquired DAC serial conversion module, also provided by Dorji.

Since these modules are specially designed for sensor type applications, it gives us some predefined options to what kind of communication will be established. The one required for this purpose is the Data Transmission Mode and it was configured with the parameters show in Figure 47 . Other modes and programming parameters are further explained in Appendix B.1.

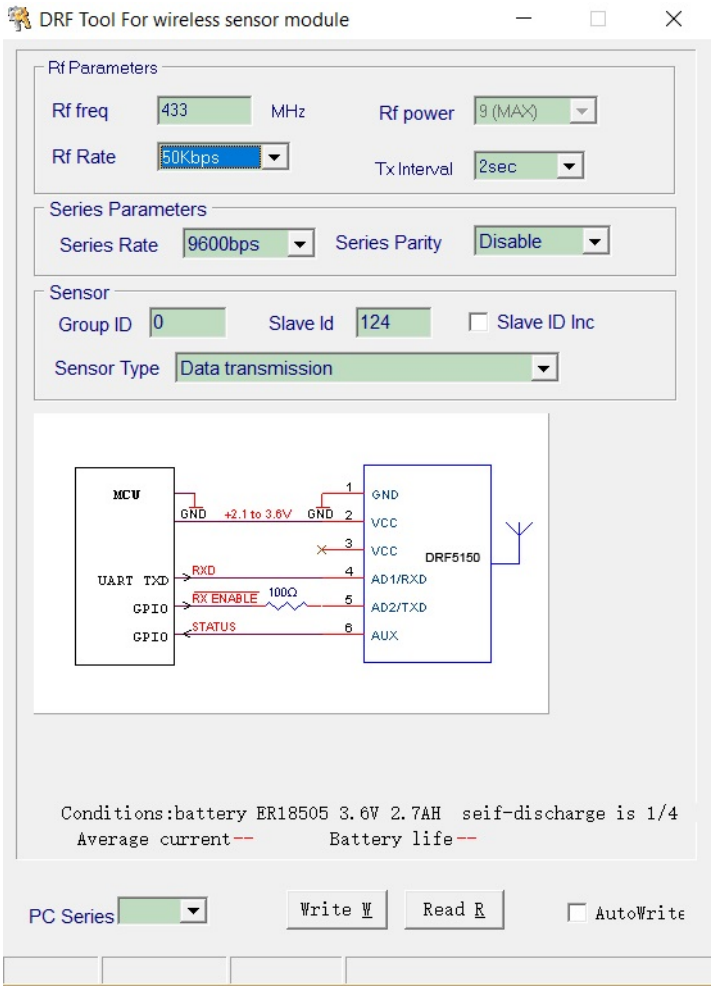


Figure 47: DRF Tool 5150 Data Transmission Configuration

4.9.3 MATLAB

MATLAB software was used on two different ends. The first one was a test to the IoT modules, as to better understand how they work and perform some coverage and range tests and the second was a script that retrieves spectrum data from the IoT receiver, processes and displays it.

GUI Sensor Test

For testing the modules a specific temperature sensor (DS18B20) was used, since the DRF Tool has a dedicated mode for that sensor.

A user interface was developed to operate this sensor using GUIDE in MATLAB. In this GUI it's possible to monitor the temperature in real time from different sensors, as well as their battery life and signal strength, thus it can further plot the temperature or the RSSI in real time. In figure 48 we can see an image of GUIDE and the prototype of this interface.

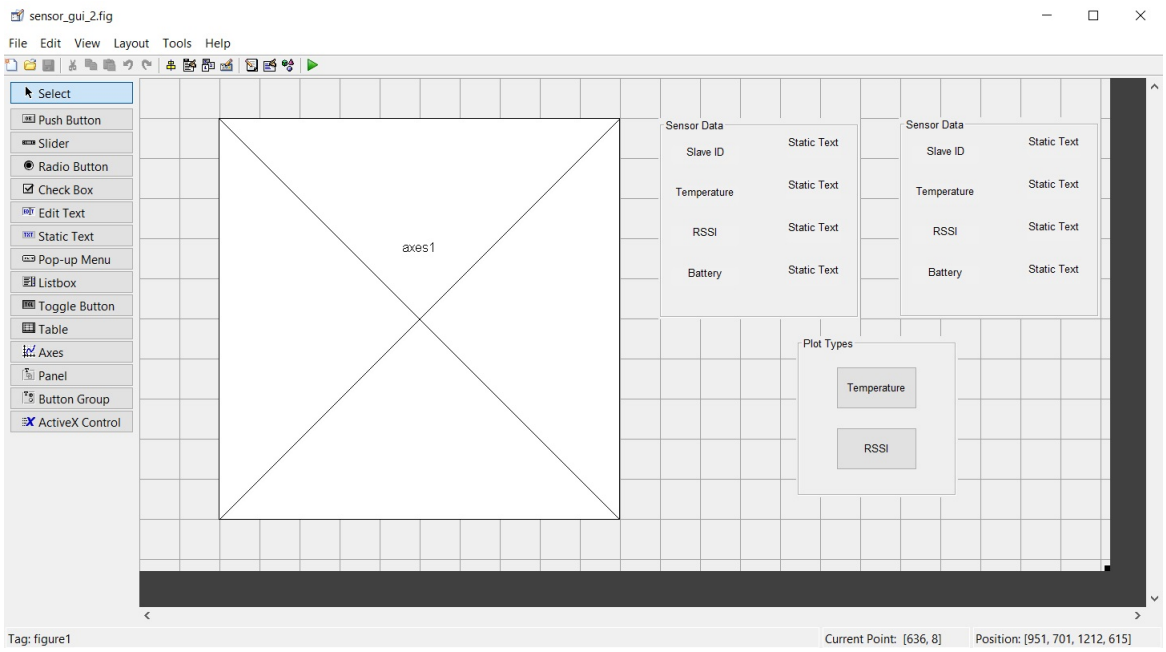


Figure 48: DS18B20 Temperature Sensor GUI Prototype

This communication with the IoT receiver module was made through UART protocol directly implemented in MATLAB.

The development of this GUI further helped to prepare for an eventual GUI for the spectrum sensor which may be implemented in future work. All the code regarding this application can be found in appendix B.1.

Spectrum Sensor Data Processing

This is the end user application for the processing and visualization of the spectrum data. Its algorithm is simple, the following flow chart describes it:

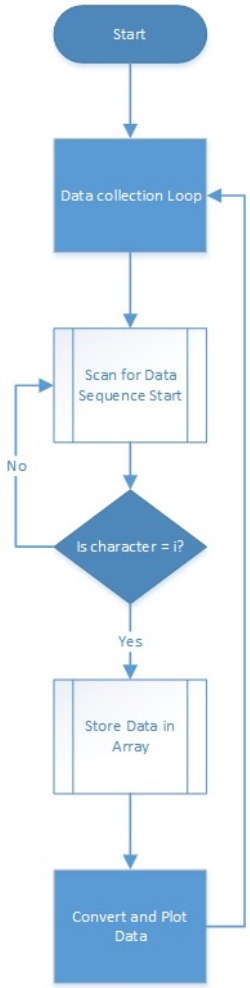


Figure 49: Spectrum Data Collection Algorithm

Upon starting the program the script searches for a serial port object, which will be the DRF4432S transmitter, once the pairing is done it opens the device.

A loop of data collection is initialized, inside that loop a start condition for the sequence of data must be applied, because the sensor is always sending information and the script could "catch" that information in any random order. For that purpose on the Arduino IDE side an extra character is sent to flag the start of a sequence.

In the MATLAB terminal that character is scanned to indicate the start of a sequence of data, from that point, sequential scans are performed to collect the desired data, which are the frequency scanned, the VGC1 valued applied and the DC output of the signal.

The data is stored in arrays and are then finally converted to the proper type and then plotted to the user.

Chapter 5

Results and Analysis

5.1 IoT modules Test and Results

Since there wasn't many information online, a contact was made with Dorji's Support and they indicated the tutorial in [41]. In order to test the IoT modules, and better understand their way of functioning, a test was made with a specific temperature sensor DS18B20 from Dallas Semiconductor [42] as referred in the tutorial. Image 50 shows the independent sensor powered by two AA batteries which provide in average 3.15 V.

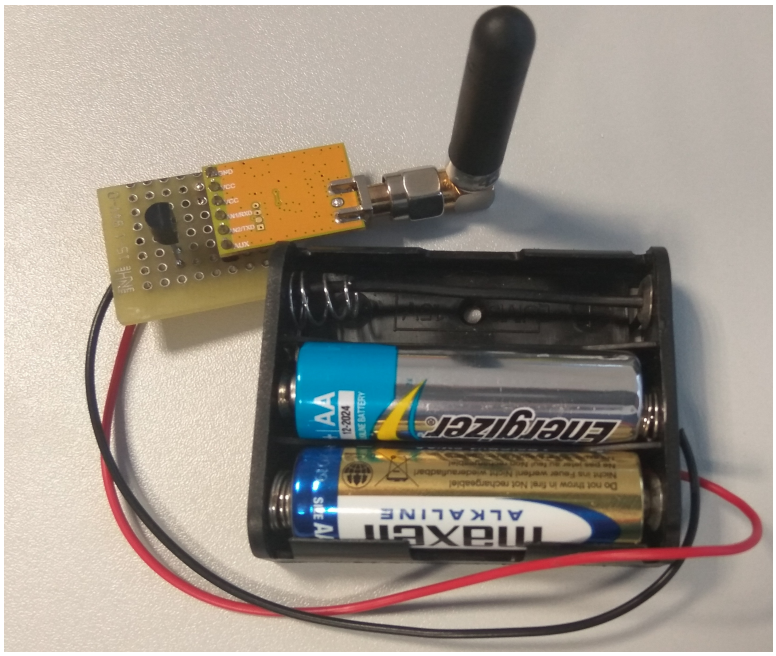


Figure 50: DS18B20 Temperature Sensor with DRF5150S

5.1.1 DS18B20 GUI Test

As stated in subsection 4.9.3, in order to prepare for an eventual GUI in the end user application for the Spectrum Sensor, a GUI was made through GUIDE in MATLAB in which the Slave ID, Temperature, RSSI and Battery can be monitored in real time, figures 51 and 52 show the GUI with real time plot of each parameter.

In the GUI it's possible to change the type of plot being seen and to know any point of the plot by clicking the fourth symbol on the upper menu.

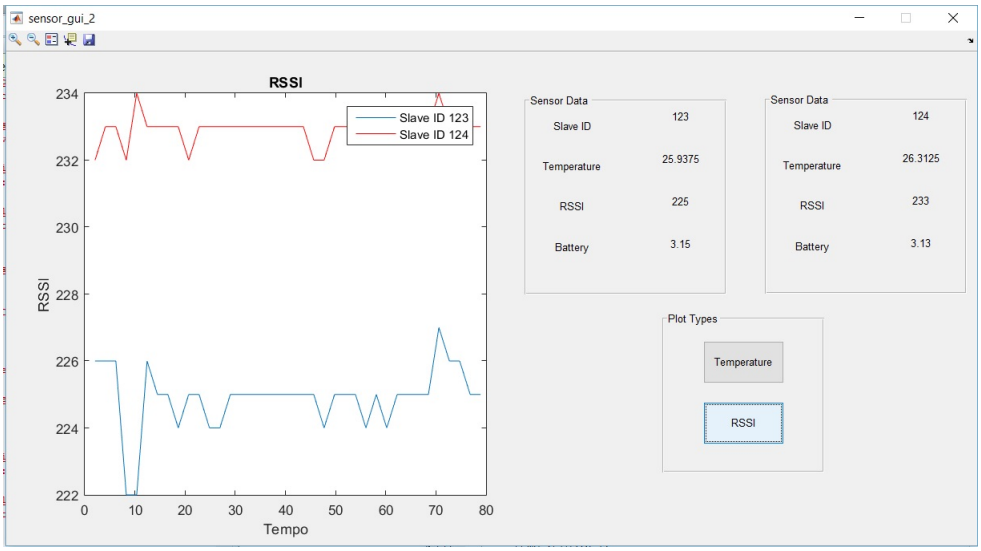


Figure 51: DS18B20 Temperature Sensor GUI - RSSI Data

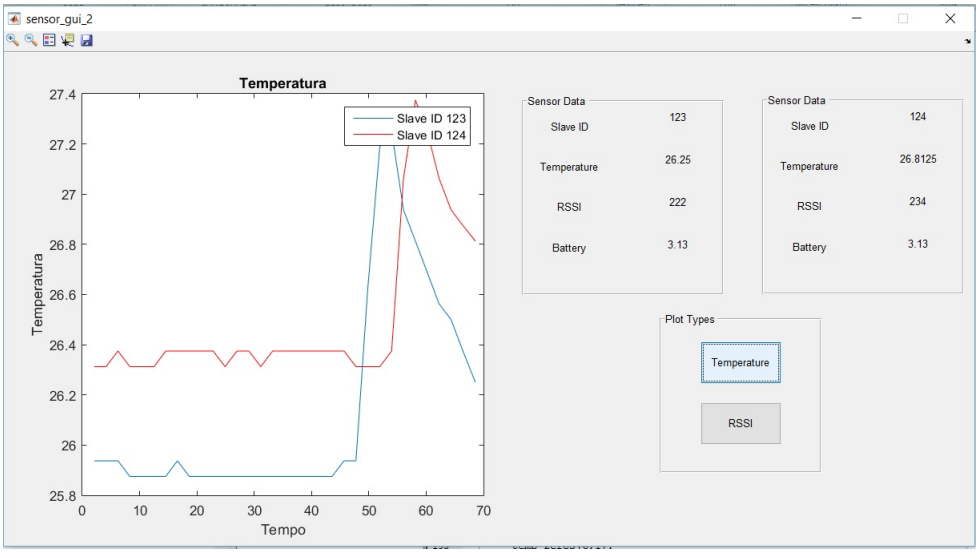


Figure 52: DS18B20 Temperature Sensor GUI - Temperature Data

5.1.2 Range Test

Initially a test was made inside the IT department, in which the modules covered only half of the department's are, however this is not the best location for testing the modules since it's a facility with major communication interference, and of course the modules are supposed to work outside so a test was performed outdoors. To test how the communication link holds according to the distance from the sensor, tests were made varying the Data Rate and Resolution Mode. To measure the distance equally spaced steps were taken to increase the distance, for this case a step will be approximated to 1 meter, the distance was measured in a straight line.

Several measures were performed for THREE data rates at High Resolution Mode and the last one with repeated data rate but changed to Low Resolution Mode. The results were as follows:

Figure 53: RSSI in order to distance in High Resolution Mode @ 50 kbps

Figure 54: RSSI in order to distance in High Resolution Mode @ 25 kbps

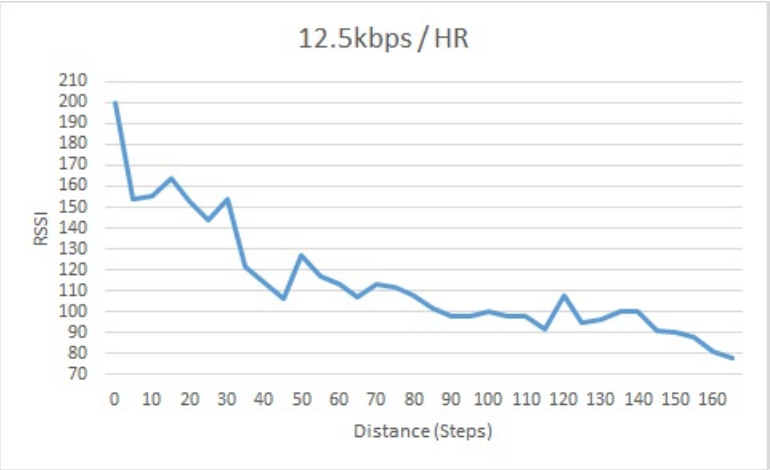


Figure 55: RSSI in order to distance in High Resolution Mode @ 12.5 kbps

Figure 56: RSSI in order to distance in Low Resolution Mode @ 12.5 kbps

With a first glance at graphics 53, 54, 55, 56, one thing stands out which is that no relevant distance increase was verified.

We would expect that the lower the data rate, a longer distance could be held, but the difference is minimal, from the fastest configuration (50 kpbs) to the slowest (12.5 kbps) the difference is no more than 30 to 35 steps (or meters).

As for the resolution mode, there was also no relevant increase of distance from High to Low, even though in Low Resolution the measurements update a lot faster.

A note to be made, is that the RSSI doesn't go below 75 because the communication link is lost below that, in fact within the last measurements the signal was lost several times, which means that some factor was obstructing the signal.

There are, although, a great number of factors that can directly influence these measurements, one of them is communication interference, others can be partial obstruction of the line of sight and radiation out of the Fresnel zone.

To reach the ideals of communication, the modules would have to be tested attending the Friis Formula 13 and the Fresnel Zone.14

$$P_r = G_t G_r \frac{\lambda}{4\pi R}^2 P_t \tag{13}$$

$$r = \sqrt{\frac{N\lambda d_1 d_2}{d_1 + d_2}} \tag{14}$$

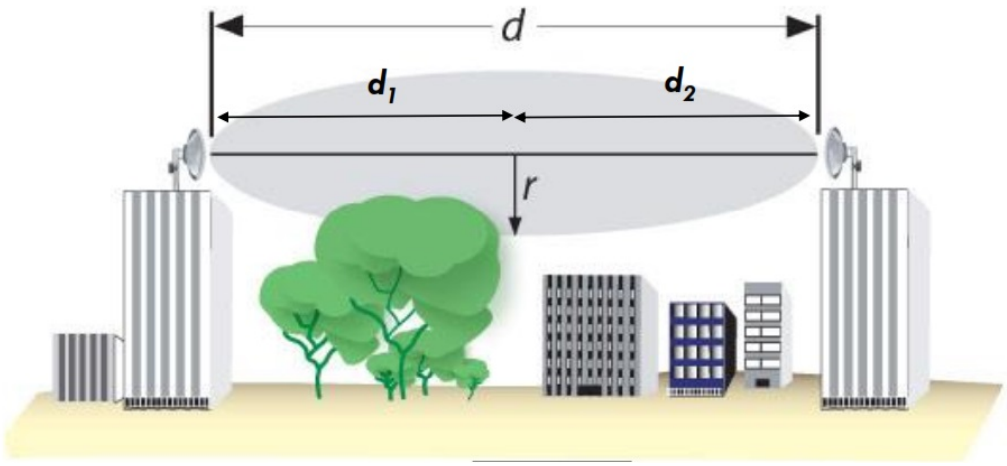


Figure 57: Ilustration of the Fresnel Zone [43]

5.2 MAX2112 Chip Characterization

5.2.1 Output I/Q Signal

To start with, a set of measurements of the amplitude of the I/Q signals were taken at different frequencies in the dynamic range of the chip for the I and Q output signals, Table 1 and figure 58 show the results for the various frequencies. These measurements are important for calibration purposes.

fLO(MHz)	I(V)	Q(V)
925	1.21	1.18
1025	1.15	1.17
1125	1.12	1.12
1225	1.12	1.12
1325	1.04	1.04
1425	1.00	1.00
1525	0.96	0.96
1625	0.92	0.92
1725	0.8	0.8
1825	0.65	0.635
1925	0.786	0.771
2025	0.8	0.8
2125	0.8	0.8
2175	0.752	0.766

Table 1: I and Q measurements for different frequencies

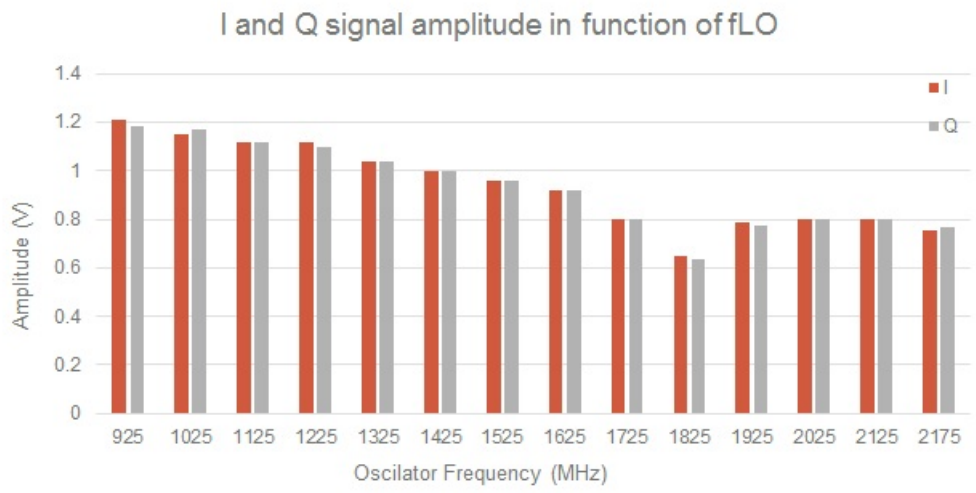


Figure 58: Variation of I / Q amplitude (Peak to peak) as a function of Oscillator Frequency

These measurements were performed for a BBG = 0 dB, VGC1 = 1.3 V and Pin = -20 dBm (the input power is set on the signal generator).

According to graphic 58 we can easily see the amplitude is not the same over all the frequency span, this is possibly due to the variation of the Input Return Loss (59) extracted from the datasheet [34] that influences the overall output signal.

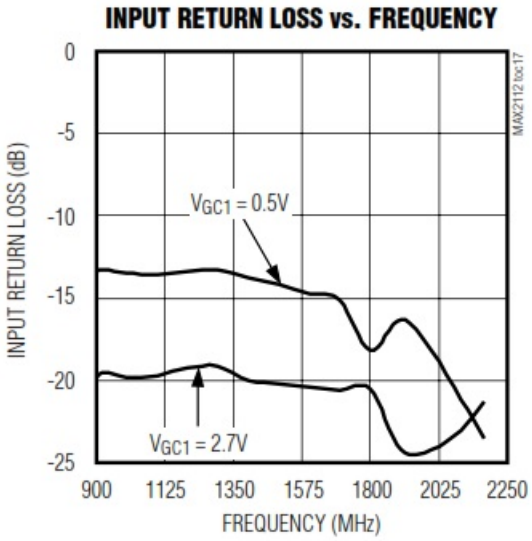


Figure 59: Input Return Loss vs Frequency

To add up to these measurements figures 60 and 61 show a print screen of the oscilloscope when measuring the signal at 1025MHz and 2025 MHz respectively, the other parameters were as follows: VGC1 = 1.8 V and BBG = 15 dB at -20 dBm.

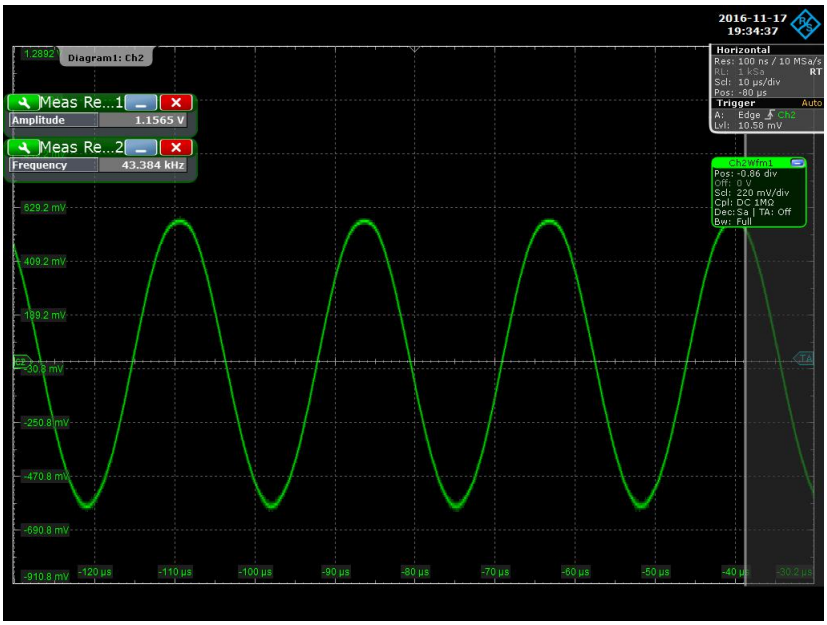


Figure 60: Oscilloscope amplitude and frequency measurement of Q signal @ 1025 MHz

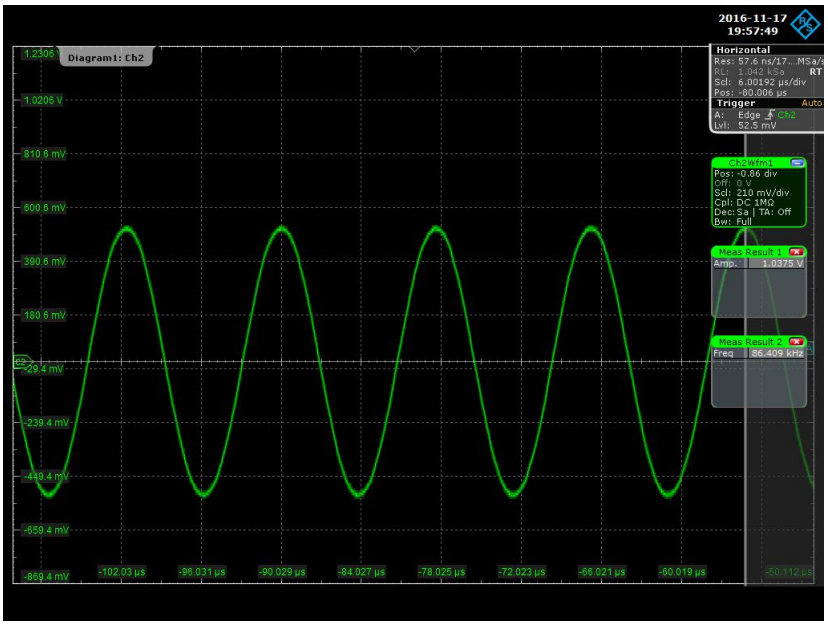


Figure 61: Oscilloscope amplitude and frequency measurement of Q signal @ 2025 MHz

One thing stands out from these results, and that is the fact that the output frequency is not exactly Baseband but a close to 0 IF, since it goes around 40 and 80 kHz. The saturation of this signal happens around approximately 1.2 V, even though that is not visible in these graphs.

5.2.2 LPF Bandwidth Test

For this test, the LPF register within the MAX2112 configuration registers, LPF[7:0], was programmed to its minimum value which is 4MHz, all the remaining configurations are the same as the previous test for the I/Q amplitude. Filter behavior was measured for three frequencies corresponding to the extremes and middle of the whole span. Test conditions are equal to the ones in section 5.2.1.

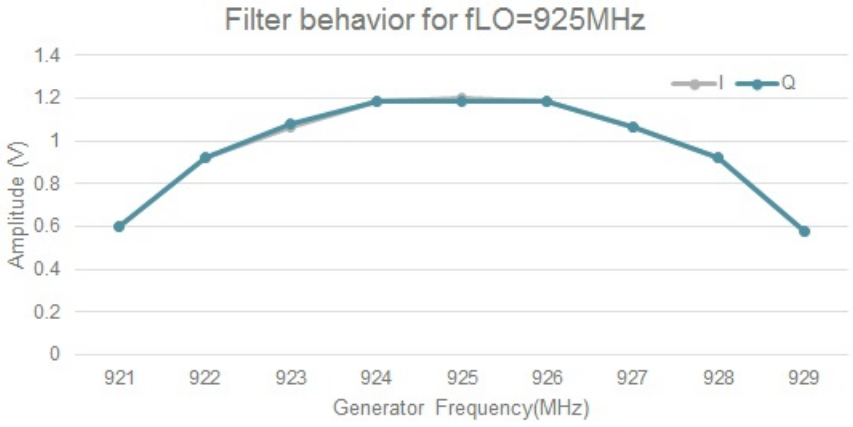


Figure 62: Filter response for 925 MHz

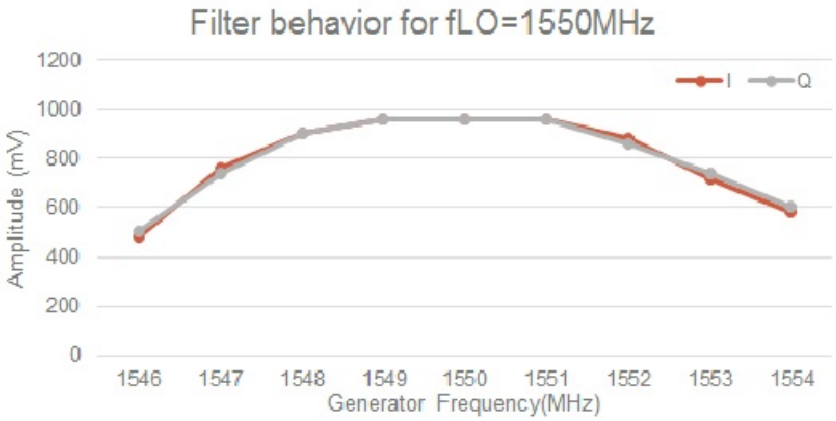


Figure 63: Filter response for 1550 MHz

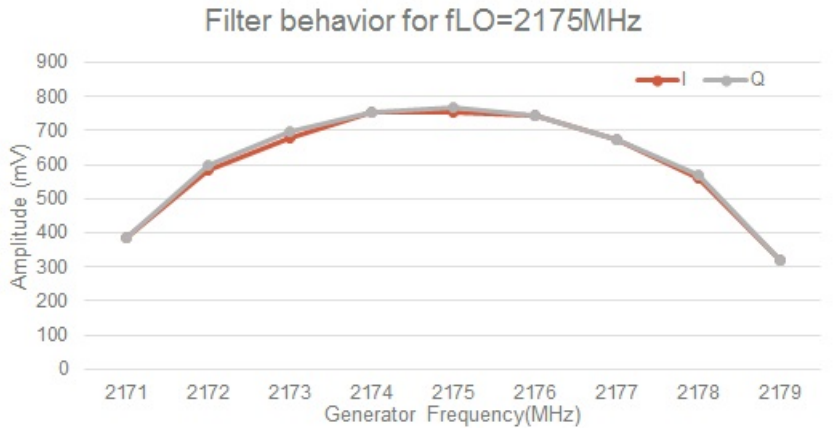


Figure 64: Filter response for 2175 MHz

From the graphics above, the behavior of the filter seems pretty accurate for the intended Bandwidth.

5.3 Power Conversion

To perform the Power Conversion (i.e. the conversion of the output of the power detector into the approximated value of the real input power), in MATLAB, the system needs to be very well characterized. The whole system has some main blocks that can introduce significant losses to the output, as we can see through the diagram in figure 33, the main elements that introduce loss, are the BBG amplifier, the RF Gain Controller, the mixer, the LPF and, externally to the MAX2112 chip the Power Detector block.

Unfortunately some of these losses aren't known, and they cannot be measured efficiently, because as we've seen in figures 60 and 61, the output goes around frequencies between 40kHz and 80kHz, and neither the laboratory's power detector, or the spectrum analyzer can measure efficiently at those frequencies, and even if it could, some of them couldn't be measured either way.

The chosen solution to reach an approximate value of the Input Power is to characterize the system, without the Power Detector connected for the chip's dynamic range (0 to -75 dBm), subtract the chosen VGC1 value in terms of gain, and then characterize the Power Detector for those powers. Of course this will only provide trust worthy results for a range of around 30 to 35 dB of power.

5.3.1 RF Gain Control

First off, we should establish the relation between tunning voltage and corresponding gain or attenuation. In page 15 of the datasheet [34] it's stated that the variable gain LNA provides 73dB of RF gain range, with 0.5V corresponding to minimum attenuation and 2.7V to maximum attenuation. However some tests revealed that when to close to the voltage limits, the relation is not so linear, in fact from 0.5 to 0.7 V and 2.5 to 2.7 V, there is almost no variation in the signal whatsoever. Therefore the following linear approximation was assumed in graph 65.

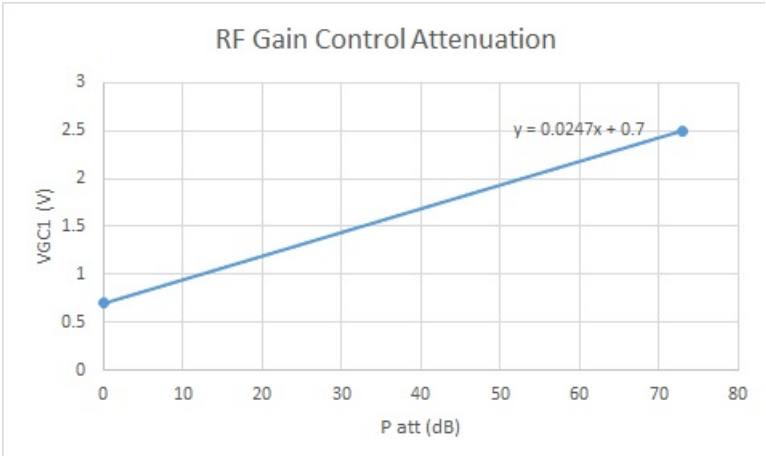


Figure 65: Gain variation with control voltage in the RF Gain Controlled LNA

5.3.2 MAX2112 Pin vs Pout

To calculate the output power of the MAX2112, an oscilloscope probe was placed at the I/Q output with a 50 Ω terminator. For constant values of BBG = 15 dB and VGC1 = 1.8 V the value of the RMS voltage was taken, this value applied to equation 15 resulted in the Power values in graphic 66.

$$P_{out} = \frac{V^2}{R} \tag{15}$$

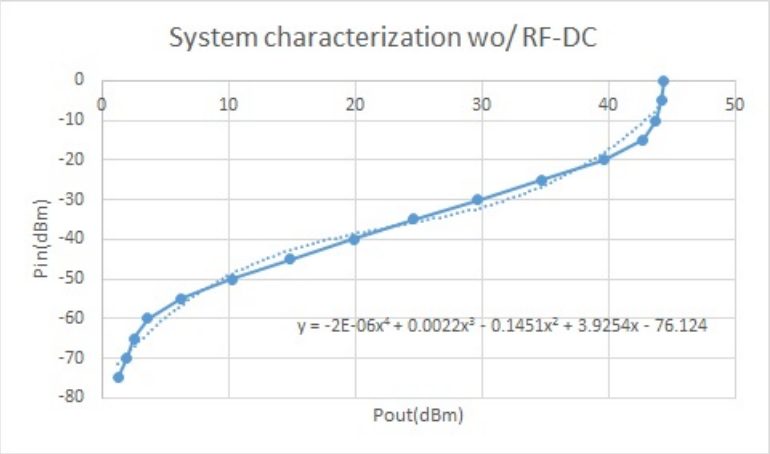


Figure 66: MAX2112 Characterization for Pin vs Pout

As we can see in graphic 66 the results are only linear from around -20 to -60 dBm. The -20dBm limit is due to the signal's saturation and the -60dBm limit is due to insufficient signal amplitude. The best equation that defined this correlation was a 4th order polynomial approach, either above or below that, the results become very inaccurate.

Up next we proceed to characterize the Power Detector.

5.3.3 Power Detector

The Power Detector outputs a DC voltage proportional to the amplitude of the captured signal. To know what power corresponds to a certain voltage a test was performed using a Vector Signal Generator directly into the input of the detector, as we change the power level on the generator we observe the corresponding voltage. The power range in which the power detector was evaluated is the Pout from 66 minus the corresponding VGC in dB calculated from 65 which is approximately 43dB. The results are in the graphic below.

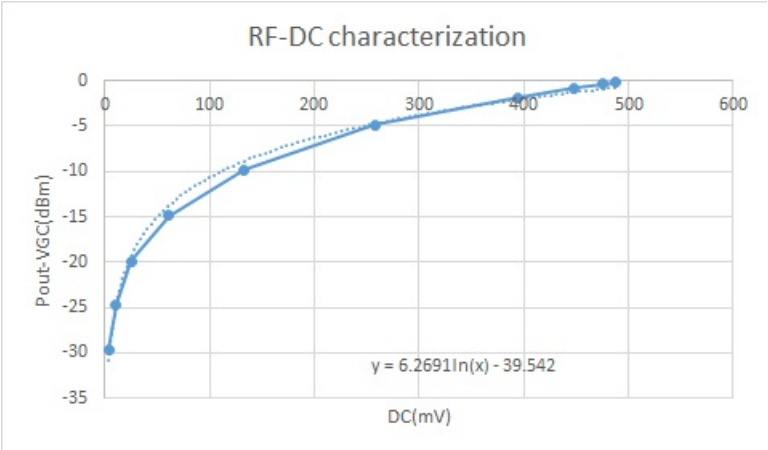


Figure 67: Relation between the DC output and output power of the MAX2112 without gain compensation

Giving the curve obtained the best approach to this relation would be a logarithmic approximation. Once again we can see that the range of this power detector is also limited. A DC value below 50 mV will also be inaccurate.

below are two tests performed with the same parameter settings of 60 and 61:

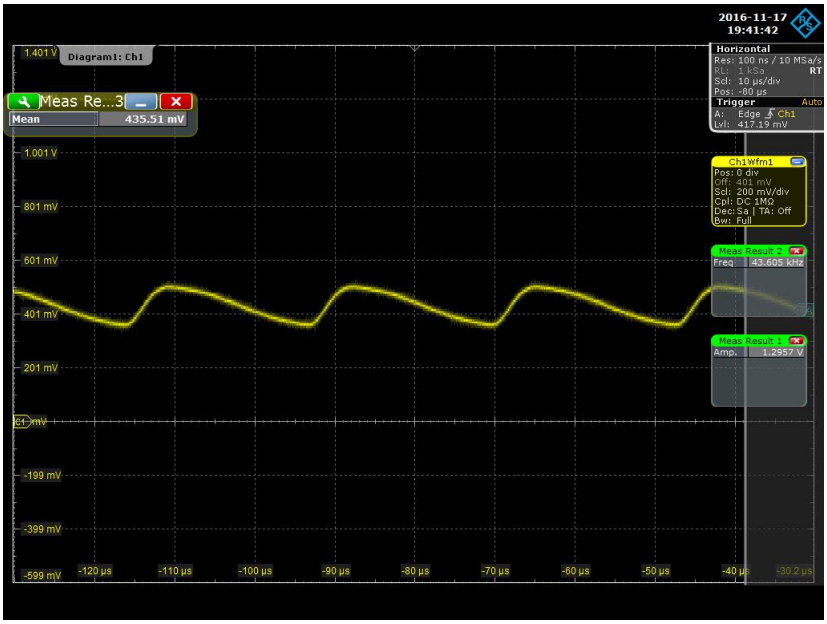


Figure 68: DC output voltage for 1025MHz

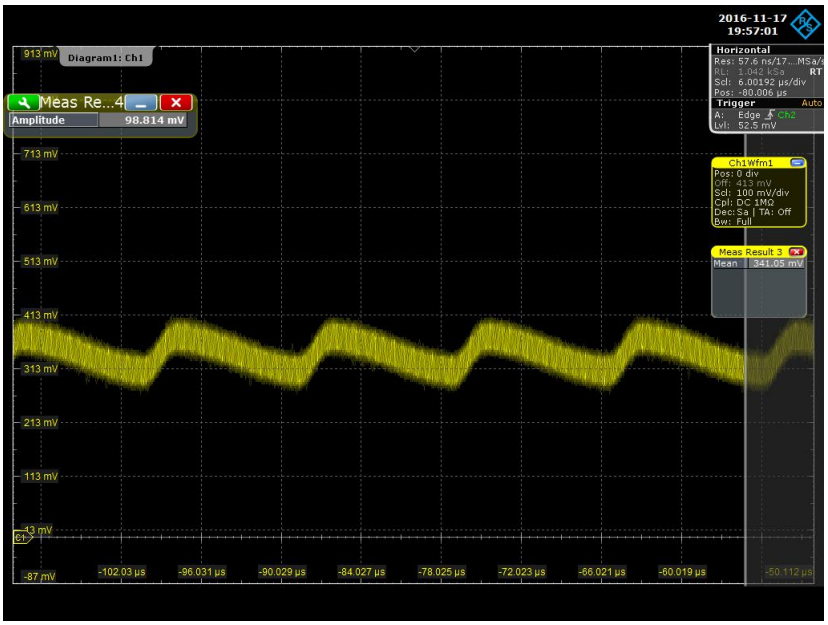


Figure 69: DC output voltage for 2025MHz

The DC values taken off these graphics are respectively 435.51 mV and 341.05 mV in average, but they are not very reliable, because this test was done without taking in consideration the linear range of DC values for the detector, if we applied the conversion method the results for the input power would be around -7 dBm which is not very accurate, comparing to the actual -20 dBm applied.

The distortion present in figure 69 is due to poor wiring (before the accommodation of the wires) at the moment of this measure, since it was highly sensitive to the touch.

As we can see, in the worst case scenario we have a maximum ripple of approximately 100 mV, this also depends on the applied voltage in VGC1: the higher the VGC1 generated, the higher the ripple will be.

5.3.4 Frequency Sweep

For this test an RF signal at 935 MHz was used as input to the system, at 3 different input powers, -30, -40 and -50 dBm, which is the range in which the system works best for the current tuning. The results were processed and plotted by MATLAB and they're shown in the following figures:

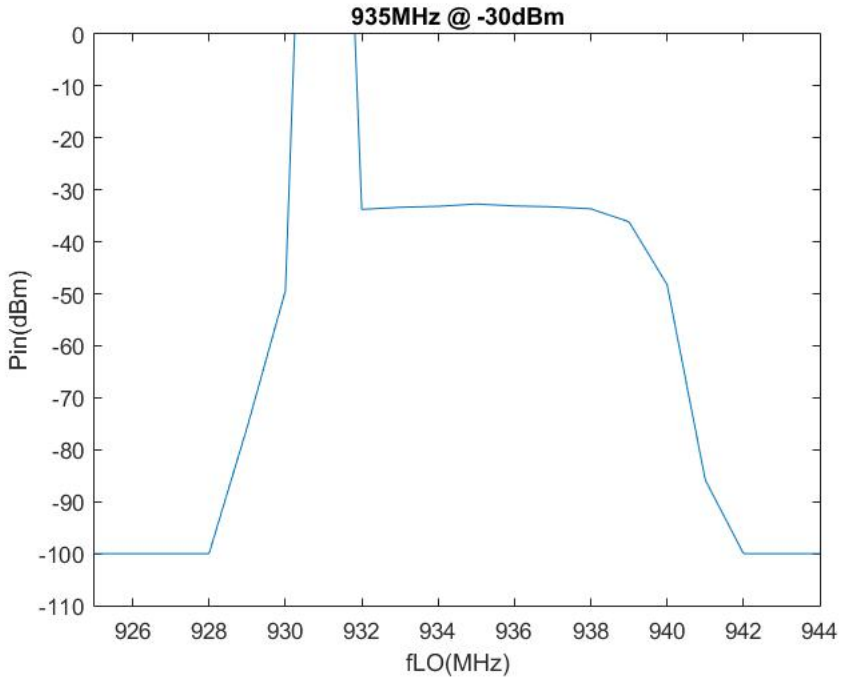


Figure 70: Results for frequency sweep with signal at 935 MHz @ -30 dBm

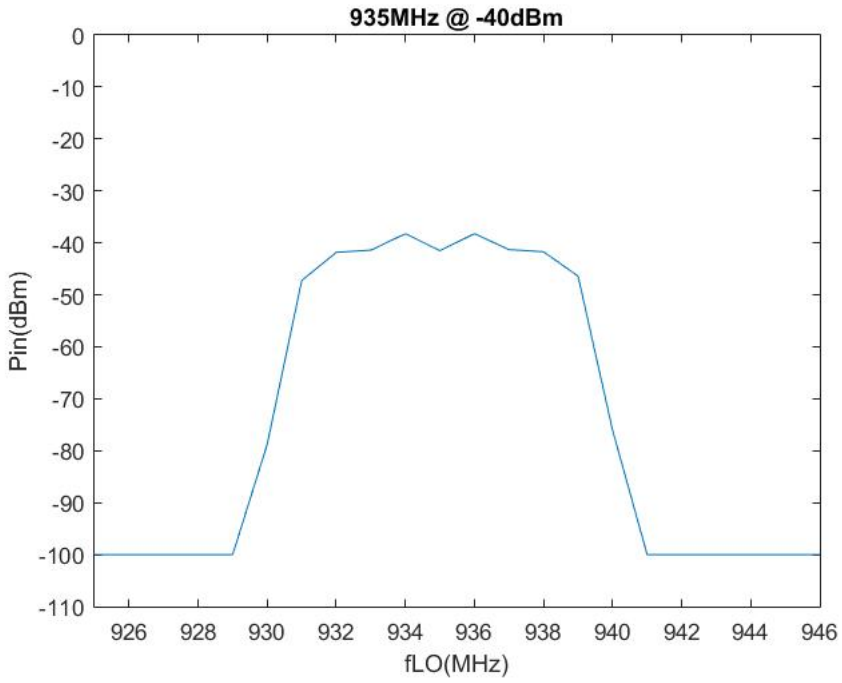


Figure 71: Results for frequency sweep with signal at 935 MHz @ -40 dBm

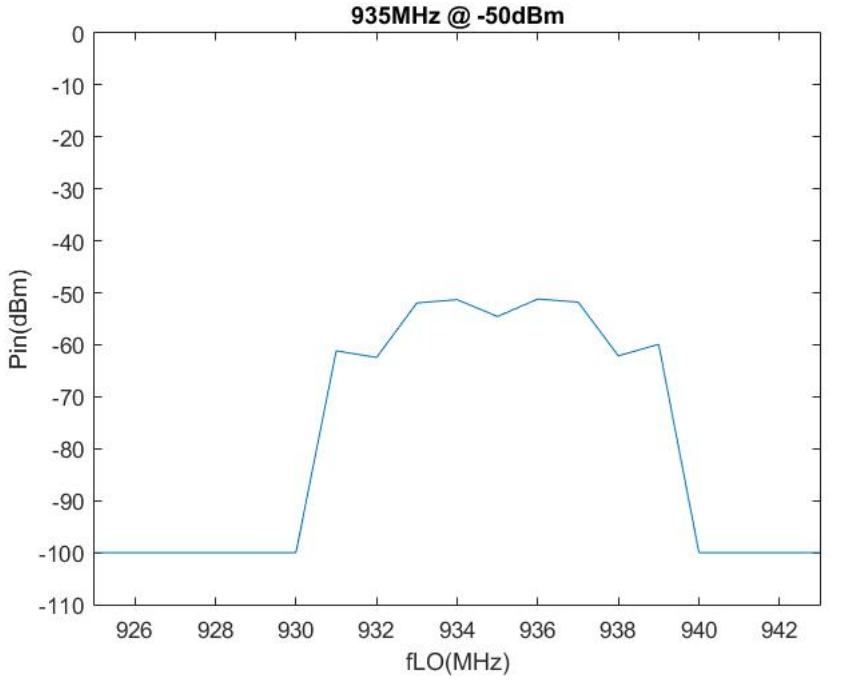


Figure 72: Results for frequency sweep with signal at 935 MHz @ -50 dBm

First of all we should explain the observed results in general. The reason why we see a signal more or less the same power for 4 frequencies before and after the desired frequency is because of the LPF setting which is set to 4MHz and it's the minimum possible.

Secondly in the first plot in figure 70 we observe a power overshoot four frequencies below the intended one it happens for any power above -30dBm, and it's probably related to the disparity between frequencies. It's not a conversion error since it already comes from the returned DC value of the power detector during the gain control sweep, its value isn't shown because being too high it would make the data that matters incomprehensible and the value itself doesn't make sense.

At last, analyzing how much the calculated power goes near the original one, the results are very close to the expected, although, and this is very important, this result is highly affected by the charge level on the batteries, at this point the batteries were approximately on 4.3 V, but if by any reason the voltage lowers, this will affect the applied voltage on the RF Gain Control.

This variation is due to the way the ATMEGA is coded, in the calculation of the applied voltage one must convert a value from 0 to 255 to a value between 0 and the supply voltage of the ATMEGA328P, this supply is not constant in this case, but variable since we're using AA batteries, thus the variation in the results.

Chapter 6

Conclusion

The developed work in this dissertation had the main goal of creating a functional spectrum analyzer sensor with the largest dynamic frequency range as possible between 30 MHz and 6 GHz. Even though the achieved range doesn't cover even half of this interval and ended up with 925MHz to 2175MHz it's still functional low power and low cost spectrum sensor within this range, and covers some of the intended communication protocols.

Although the system can be successfully configured to detect signals from 0 to -75 dBm, which is the system's dynamic range, the Power output for the user is not accurate to the entire range, but only to about 30 dBm within this range. This is mainly due to limitations of the power detector and lack of time to properly evaluate the system in all ways possible. For totally accurate results for all the dynamic range one would have to take in account the variation of the signal's amplitude to the entire frequency range, the behavior of the system to the different VGC1 values, and the power losses along the circuit.

The IoT component of this project was also achieved, in which two modules successfully communicate the received spectrum data, despite having a not so high range of coverage. The accuracy of the system is not very high, due to the low quality of the filter used to apply the VGC1 voltage, that can lead to some errors.

Summing up, it was proven that it's possible to implement a Spectrum probe for the IoT, based on SDR techniques,and since this project is subjected to a lot of improvements I truly believe that an extremely efficient and accurate system is possible to achieve at low cost and power.

6.1 Future Work

There is room for a lot of improvements in this system.

A major improvement would be to have a decent conversion of the captured power regarding the frequency range and successfully knowing which power is being detected in the sensor for all the frequencies in all of the possible conditions with sufficient accuracy.

Also since this is supposed to be an independent, low power portable device it would be nice to implement an energy harvesting feature to make it self sufficient, using for example the sunlight to power the system. Thus, above that, the power source should be stable, to avoid errors the situation referred in 5.3.4, a different approach would be to design a system that measures the supply voltage continuously and applies that value to the calculations in the code, but it would be a waste of resources.

As shown by the tests with the temperature tests it's possible to enhance this spectrum sensor to be more than a spectrum sensor, adding temperature, humidity, pressure sensors or any kind of sensors, is totally feasible and adds up to the IoT knowledge.

As for the internal system itself, it would be good to improve the performance of the LPF used to apply the DC voltage to VGC1, leading to less errors and more accurate results. I also believe the code itself can be optimized, one just has to experience with delays and other techniques to allow having better and faster results.

In a perspective of power savings, better MCU ' s exist out there that consume less than the Arduino, this was just the chosen platform for this work and making it easier to implement.

The power detector also could be replaced with a market professional chip instead of a custom one subjected to characterization and limitations.

Regarding the communication end, one big improvement that can be made is to use transceivers instead of straight forward transmitter / receiver. This would allow for a bi directional communication to take place and therefore, the remote reprogramming of the device without having to retrieve it from its medium, which also adds up to the concept of ideal SDR and intelligent radios. This would also allow for a more intuitive GUI to be created in which the user can easily reprogram the sensor to his best interest.

6.2 Difficulties

For anyone who has some interest in picking this project where it was left on, or maybe even build their own, this section is dedicated to provide some advise, so you don't lose as much time as I did in some fields.

Regarding the RF front end: do your research, chose the one that best suits your needs, be sure the datasheet has enough clear information for you to carry on with its programming easily, but most of all when building the application circuit for the chip, build it yourself and avoid using existing ones, because if you don't know how it was built, and if it has errors you'll take a long time to figure it out, like it happened with me.

One main issue that I found was that very few information was available online either for the RF front end or the IoT modules, so be sure to take the information available factor in mind when choosing the components.

Something that also made me lose some precious time was digging into the IoT modules programming. I was expecting to have to program with code and kept searching for Source codes for my modules with no success, when in fact the only available option was the software tool to automatically program it. So if you really want to program your modules from scratch be sure to chose ones that really have that option easily accessible.

As it's known in this work a custom power detector was used, a not very accurate one, that still had to be characterized and studied, of course it also has some drawbacks in the dynamic range, but a better market chip can be bought that already has its behavior laid out for you on the datasheet.

Bibliography

[1] Quadro Nacional de Atribuição de Freqências. Online in http://www.anacom.pt/streaming/qnaf06_versao_int.pdf?contentId=313152&field=ATTACHED_FILE. Consulted in September 2016.

[2] The Evolution of Mobile Technologies - Qualcomm, 2014 Online in https://www.qualcomm.com/media/documents/files/the-evolution-of-mobile-technologies-1g-to-2g-to-3g-to-4g-lte.pdf. Consulted in October 2016.

[3] Measuring with Modem Spectrum Analyzers - Educational Note Online in https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/1ma201_1/1MA201_9e_spectrum_analyzers_meas.pdf. Consulted in September 2016.

[4] N.B. Carvalho and D. Schreurs "Microwave and Wireless Measurement Techniques". Cambridge, 2013.

[5] Spectrum Analyzer Basics Tutorial - Radio-Electronics. Online in http://www.radio-electronics.com/info/t_and_m/spectrum_analyser/rf-analyzer-basics-tutorial.php. Consulted in October 2016.

[6] J.C.S Correia "Implementacão em hardware de um analisador de espectros baseado em SDR", Masters' Dissertation, DETI, UA, Aveiro, PT, 2010.

[7] Application Note 150 - Spectrum Analysis Basics - Agilent Technologies Online in http://cp.literature.agilent.com/litweb/pdf/5952-0292.pdf. Consulted in September 2016.

[8] Application Note 150-15 - Vector Signal Analysis Basics - Agilent Technologies Online inhttp://cp.literature.agilent.com/litweb/pdf/5989-1121EN.pdf. Consulted in October 2016.

[9] J. Mitola. "The Software Radio Architecture", IEEE Communications Magazine, 1995.

[10] N.B. Carvalho, A. Cidronali, R. Gomez-Garcia "White Space Communication Technologies". Cambridge University Press, 2015.

[11] Air-to-Ground Radio based on SDR-4803 Embedded Radio Module. Online in http://www.spectrumsignal.com/custom-cots-service/. Consulted in October 2016.

[12] P. Cruz and N.B. Carvalho, Paper on “Multi-Mode Receiver for Software Defined Radio". IT, UA, Aveiro, PT.

[13] > 50Gsps CMOS ADC, DAC and DSP - Fujitsu Online in https://indico.cern.ch/event/121657/attachments/68435/98170/ADC_DAC_CERN.pdf. Consulted in October 2016.

[14] Software Defined Radio. Online in http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf. Consulted in October 2016.

[15] Cognitive Radio Tutorial - Radio - Electronics. Online in http://www.radio-electronics.com/info/rf-technology-design/cognitive-radio-cr/techn. Consulted in October 2016.

[16] P. Cruz, H. Gomes and N.B. Carvalho, “Receiver Front-End Architectures – Analysis and Evaluation, Advanced Microwave and Millimeter Wave Technologies Semiconductor Devices Circuits and Systems". IT, UA, Aveiro, PT, 2010.

[17] J.M.M.L. Mendes, “Security Techniques for the Internet of Things". DETI, UA, Aveiro, 2013.

[18] F. Xia, L.T. Yang, L. Wang and A. Vinel, “Internet of Things", International Journal of Communication Systems, 2012.

[19] S.M. Ghaleb, S. Subramaniam, Z.A. Zukarmin and A. Muhammed, “Mobility Management for IoT: a survey", EURASIP Journal on Wireless Communications and Networking, 2016.

[20] F.J. Jariego, “Industrial IoT Chile, 2014". Online in http://www.slideshare.net/FranciscoJariego/iot-46085358/5. Consulted in October 2016.

[21] Amsterdam Internet Exchange - IoT Survey. Online in https://datafloq.com/read/internet-of-things-more-than-smart-things/1060. Consulted in October 2016

[22] Novotny R, Kuchta R, Kadlec J, “Smart City Concept, Applications and Services. J Telecommun Syst Manage". 2014.

[23] G. Margelis, R. Piechocki, D. Kaleshi and P. Thomas, “Low Throughput Networks for the IoT: Lessons Lerned From Industrial Implementation". MVB, School of Engineering, University of Bristol, UK, 2015

[24] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, “Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios" IEEE Wireless Communications, Vol. 23, Oct. 2016.

[25] Ingenu Online in http://www.ingenu.com/ . Consulted in October 2016.

[26] Extreme Range Links: LoRa 868 / 900MHz SX1272 LoRa module for Arduino Waspmote and Raspberry Pi. Online in https://www.cooking-hacks.com/documentation/tutorials/. Consulted in February 2016.

[27] Link Labs - 14 LoRa FAQs Answered. Online in http://www.link-labs.com/lora-faqs/. Consulted in October 2016.

[28] LoRa vs LTE-M vs Sigfox - Creative Connectivity. Online in http://www.nickhunn.com/lora-vs-lte-m-vs-sigfox/. Consulted in November 2016.

[29] How to choose the best connectivity network for your project. Online in https://publisher.opensensors.io/connectivity. Consulted in October 2016

[30] L. Atzori, A. Iera and G. Morabito "The Internet of Things: A Survey", Elsevier, 2010

[31] LoRa SX1272/73 transceiver modules - Semtech Corporation. Onilne in http://www.semtech.com/images/datasheet/sx1272.pdf.

[32] DRF4432S ISM RF Sensor Receiver Module - Dorji Applied Technologies. Online in http://www.dorji.com/docs/data/DRF4432S.pdf.

[33] DRF5150S Wireless Sensor Transmitter Module - Dorji Applied Technologies. Online in http://www.dorji.com/docs/data/DRF5150S.pdf.

[34] MAX2112 Complete, Direct-Conversion Tuner for DVB-S2 Applications - Maxim Integrated. Online in https://datasheets.maximintegrated.com/en/ds/MAX2112.pdf.

[35] MAX2112 Evaluation Kit Datasheet - Maxim Integrated. Online in https://datasheets.maximintegrated.com/en/ds/MAX2112EVKIT.pdf.

[36] Phase Locked Loops - National Instruments, Online in http://www.ni.com/tutorial/3781/en/. Consulted in November 2016.

[37] PLL Phase Locked Loop Tutorial, Online in http://www.radio-electronics.com/info/rf-technology-design/pll-synthesizers/phase-locked-loop-tutorial.php. Consulted in November 2016.

[38] The Basics of PLL Frequency Synthesis - Online Radio & Electronics Course Online in http://www.nsarc.ca/hf/pll.pdf

[39] ATMEL 8-BIT Microcontroller WITH 4/8/16/32KBytes In-System Programmable Flash Online in http://www.atmel.com/

[40] LT1761 Series - Linear Technology Online in http://www.farnell.com/datasheets/1563033.pdf?_ga=1.234272734.1718483785.1477672630

[41] Building Wireless Sensor Applications Using Dorji's DRF5150S and DRF4432S RF Modules. Online in http://embedded-lab.com/blog/. Consulted in February 2016.

[42] DS18B20 - Programmable Resolution 1 -Wire Digital Thermometer - Dallas Semiconductor. Online in http://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

[43] N.B. Carvalho Class 9 - Fundamentals of Propagation, Radio Systems

[44] MAX2120 /MAX2112 VAS App Note, Draft V0.2, June 27, 2007

[45] FFT Spectrum Analyzer, Radio-Electronics. Online in http://www.radio-electronics.com/info/t_and_m/spectrum_analyser/fft-analyzer.php. Consulted in October 2016

[46] T.J.M. Monteiro "Projecto de um analisador de espectros baseado em SDR", Masters Dissertation, DETI, UA, Aveiro, PT, 2010.

[47] P.M.D. Cruz "Characterization of Systems for Software Defined Radio", Masters Dissertation, DETI, UA, Aveiro, PT, 2008.

[48] L.M.M. Antunes, "Software Defined Radio em FPGA", Masters Dissertation, DETI, UA, Aveiro, PT, 2009.

[49] C. Bowick, J. Blyler and C. Ajluni, "RF Circuit Design", Second Edition, Newnes.

[50] J.M.M.L. Mendes, "Security techniques for the Internet of Things", Masters Dissertation, DETI, UA, Aveiro, PT, 2013.

[51] Smart City Concept, Applications and Services - Journal of Telecommunications System & Management. Online in http://www.omicsgroup.org/journals/smart-city-concept-applications-and-services-2167-0919-117.php?aid=33684. Consulted in October 2016.

[52] Sigfox. Online in http://www.sigfox.com/. Consulted in October 2016.

[53] LoRa Alliance Technology. Online in https://www.lora-alliance.org/What-Is-LoRa/Technology. Consulted in October 2016.

[54] LTE-M – Optimizing LTE for the Internet of Things - Nokia. Online in https://iotfuse.com/wp-content/uploads/2016/02/nokia_lte-m_-_optimizing_lte_for_the_internet_of_things_white_paper.pdf. Consulted in October 2016.

Appendices

Appendix A

Front End Programming

A.1 MAX2112 I^2C Registers

MAX2112 chip provides a whole set of programmable registers via I^2C, a table showing all these registers can be found below.

REG NUMBER	REGISTER NAME	READ/ WRITE	REG ADDRESS	MSB							LSB
				DATA BYTE							
				D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
1	N-Divider MSB	Write	0x00	FRAC 1	N[14]	N[13]	N[12]	N[11]	N[10]	N[9]	N[8]
2	N-Divider LSB	Write	0x01	N[7]	N[6]	N[5]	N[4]	N[3]	N[2]	N[1]	N[0]
3	Charge Pump	Write	0x02	CPMP[1] 0	CPMP[0] 0	CPLIN[1] 0	CPLIN[0] 1	F[19]	F[18]	F[17]	F[16]
4	F-Divider MSB	Write	0x03	F[15]	F[14]	F[13]	F[12]	F[11]	F[10]	F[9]	F[8]
5	F-Divider LSB	Write	0x04	F[7]	F[6]	F[5]	F[4]	F[3]	F[2]	F[1]	F[0]
6	XTAL Divider R-Divider	Write	0x05	XD[2]	XD[1]	XD[0]	R[4]	R[3]	R[2]	R[1]	R[0]
7	PLL	Write	0x06	D24	CPS	ICP	X	X	X	X	X
8	VCO	Write	0x07	VCO[4]	VCO[3]	VCO[2]	VCO[1]	VCO[0]	VAS	ADL	ADE
9	LPF	Write	0x08	LPF[7]	LPF[6]	LPF[5]	LPF[4]	LPF[3]	LPF[2]	LPF[1]	LPF[0]
10	Control	Write	0x09	STBY	X	PWDN 0	X	BBG[3]	BBG[2]	BBG[1]	BBG[0]
11	Shutdown	Write	0x0A	X	PLL 0	DIV 0	VCO 0	BB 0	RFMIX 0	RFVGA 0	FE 0
12	Test	Write	0x0B	CPTST[2] 0	CPTST[1] 0	CPTST[0] 0	X	TURBO 1	LD MUX[2] 0	LD MUX[1] 0	LD MUX[0] 0
13	Status Byte-1	Read	0x0C	POR	VASA	VASE	LD	X	X	X	X
14	Status Byte-2	Read	0x0D	VCOSBR[4]	VCOSBR[3]	VCOSBR[2]	VCOSBR[1]	VCOSBR[0]	ADC[2]	ADC[1]	ADC[0]

Figure 73: Table of programmable and readable registers in MAX2112

All the chosen values for the registers can be found in the Arduino Code appendix A.3. Until then this appendix briefly describes each of the registers' function.

All the registers from 1 to 5 have one main function which is to set the frequency for the frequency synthesizer. **N-Divider** registers set the bits for the PLL integer-divide number and **F-Divider** Registers set the fractional-divide number basically the first ones set major

spaced values of the frequency and the second ones allow for lesser jumps of frequency, both combined allow any frequency configuration as the user wants, the calculations are already done in the code, all we have to do is simply type in the desired frequency. In between these registers we also have two **Charge Pump** Registers which control minimum pulse width and linearity, these, however have default values of 0 and 1 that must be programmed upon powering up the device and should not be changed.

Register 6 is the **XTAL Buffer and Reference Divider** Register which control the dividers for the crystal and PLL.

Register 7 is the **PLL** Register, in this register there is one bit that's very important which is the D24 bit, it basically sets the **VCO** divider setting depending on the LO frequency, it should be set to 0 if the LO frequency is equal or greater than 1125MHz and to 1 if it's below 1125MHz. This register requires programming before all others otherwise VCO selection may fail sometimes.

Register 8 is the **VCO** Register, it controls the starting point for the VAS option or manual selection. There is an algorithm used to speed up VCO selection this is explained in appendix A.2.

Register 9 is the **LPF** Register. Here we can set the bandwidth of the filter to what we want, our chosen value for use is 4MHz, which is the minimum possible, but it can go up to approximately 74,5 MHz.

In Register 10 we find the **Control** Register, another important setting on this register are the BBG bits, that allow to set the Baseband Gain from 0 to 15 dB, we chose the keep it maxed out so the output signal has the best form possible. This register also provides a STBY bit that allows to lower the power consumption of the chip when not being used.

Register 11 is the **Shutdown** register and should not be changed from its default operation mode.

Register 12 is the **Test** Register and also shouldn't be changed from the default values, except for the TURBO bit that should be programmed to 1 after powering up the device.

Registers 13 and 14 are not programmable registers, but readable registers that provide important status notifications, through these registers it's possible to know if VCO selection was successful and if the PLL is locked or not. In sum it allows to know if the chip was programmed correctly or not.

A.2 VCO Algorithm

In page 16 of the MAX2112 Datasheet it's stated that we should refer to the MAX2112 / MAX2120 VCO Autoselect Application Note for more information on VCO selection, however this AN was nowhere to be found, so after a contact with MAXIM by email, they sent me an unpublished version of this AN. This AN has some information on the VCO lock time and an algorithm that can minimize this time. This algorithm consists of 4 points:

- Enable VAS mode by programming the VAS bit to 1 (reg 07).
- Choose VCO divider mode (bit D24 of reg 06) and calculate fvco:

```
If fLO <= 1125MHz
Program bit D24 (reg 06) to 1 (Divide-by-4 mode)
fvco=fLO*4
```

```
Else
D24=0 (Divide-by-2 mode)
Fvco=fLO*2
```

- Choose VCO seed value (VCO[4:0]) based on look up table (see Table 1 in [40]):

```
If  fvco < f2_3, then VCO[4:0]=01010 (VCO2)
Elseif fvco < f3_4, then VCO[4:0]=01011 (VCO3)
Elseif fvco < f4_5, then VCO[4:0]=01100 (VCO4)
.
.
.
Elseif fvco < f21_22, then VCO[4:0]=11101 (VCO21)
Elseif fvco < f22_23, then VCO[4:0]=11110 (VCO22)
Else VCO[4:0]=11111 (VCO23)
```

- Calculate and load N counter value (and F counter value for MAX2112). For MAX2120 the N counter LSB word must be loaded last to initiate a new VAS sequence. For MAX2112 the F counter LSB word must be loaded last to initiate a new VAS sequence.

A.3 Arduino Code

In the MAX2112_config function there are all the editable registers if you want to change some values like BBG and the LPF. The sweep frequencies can be changed in the beginning of the loop function as well as the spacing between frequencies.

```
//Autor: Gabriel Sa Pinto
//Nmec: 64034
//Description: Configures a MAX2112 Direct Conversion Module for Spectrum Analyzing for a specific frequency sweep between 925 and 2125 MHz
//Features:
//          - Programable frequency sweep. WARNING: Using the whole range will take forever!!
//          - Performs an Analog Read (w/ smoothing) of the DC output of a Peak detector designed to later calculate the input power on Matlab
//          - Automatic chip gain control (VGC1), according to DC read value (must be aprox. between 230 and 270 mV)
//          - Sends fLO, VGC1 and DC values through Serial UART to a DRF4432S Wireless Transmitter (433 MHz)


#include <Wire.h>                 //I2C library
#define MAX2112 96                //I2C hex ID code of MAX2112 chip

 int i = 0;
 int tip[4];                      //Array of characters for LoRa sending

// VCO Transition Frequency (MHz)
int f0_1 = 2135, f1_2 = 2200, f2_3 = 2275, f3_4 = 2355, f4_5 = 2445, f5_6 = 2545, f6_7 = 2660, f7_8 = 2760, f8_9 = 2770;
```

```
int f9_10 = 2865, f10_11 = 2965, f11_12 = 3070, f12_13 = 3190, f13_14 = 3330, f14_15 = 3480, f15_16 = 3640, f16_17 = 3685;
int f17_18 = 3795, f18_19 = 3915, f19_20 = 4035, f20_21 = 4170, f21_22 = 4325, f22_23 = 4500;

int digital6 = 3; //Digital Pin 3 PWM for VGC1

////////////////////////////////////// Analog read smooth parameters ////////////////////////////////////////////////////
const int numReadings = 30;     //number of readings used in the average of DC read value

int readings[numReadings];      // the readings from the analog input
int readIndex = 0;              // the index of the current reading
int total = 0;                  // the running total

int analog1 = 1;                //Analog Pin for DC read
//int DC_an;
//////////////////////////////////////////////////////////////////////////////////////////////


void setup() {
  Wire.begin();
  Serial.begin(9600);                                 //Set baudrate for Serial transmission
  delay(15);                                          //Ensure baudrate and I2C transmissions

  pinMode(digital6, OUTPUT);   // sets the pin as output

  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
    readings[thisReading] = 0;    //clear the array for the average

 ////////////////////////////////// Run Configuration Once for fLO frequency ////////////////////////////////////////////
  /*// VGC1 control
  double VGC1 = 1.8;
  double write_value = VGC1*(255/4.4);
  //int write_value = 120;
  analogWrite(digital6, write_value);

  MAX2112_config(10n025);
  int errorcode = PrintStatusBytes();

  if(!errorcode)
  {
    Serial.println("Error in configuration");
  }

  //Print diode DC out
  int DC_an = analogRead(analog1);      //Read DC value
  Serial.println(DC_an);
  double DC = (5000*(double)DC_an)/1023;
  Serial.print(DC);
  */
  //////////////////////////////////////////////////////////////////////////////////////////////
  }
```

```
69
   ////////////////////////////////////////////////// Main Loop Function
     //////////////////////////////////////////////
71 void loop() {

73   int fLO;
     double VGC1 = 2.5;                                    //Optimal Starting
    VGC1 value
75   double write_value = VGC1*(255/4.3);                    //Set VGC1
     initial value
     analogWrite(digital6, write_value);                   //Test delay -
    SUBJECTED TO CHANGE!!!
77   delay(100);
     double DC = DC_smoothing(readings, total, numReadings);   //Calculates
     average of the read DC value based on numReadings set above
79
     ////////////////////////////////////////////// Frequency Sweep
     ////////////////////////////////////////////////////
81   for(fLO = 925; fLO <1025; fLO++)
     {
83
       MAX2112_config(fLO);                                //Configure module
     for current fLO
85       delay(15);
         int errorcode = PrintStatusBytes();
87
         if(!errorcode)
89         Serial.println("Error in configuration");        //Shows error if
     configuration wans't successful

91       for(VGC1 = 2.5; VGC1 > 0.5; VGC1 -= 0.1)            //VGC1 sweep inside
      one frequency to check for relevant power spikes
         {
93         double write_value = VGC1*(255/4.3);
           analogWrite(digital6, write_value);
95         delay(200);
           //Serial.println(VGC1);                           //Debug VGC1
     value
97         DC = DC_smoothing(readings, total, numReadings);  //Write VGC1 value

99
           if(DC > 4.0)                                    //Threshold for
     relevant power values
101        {
             //Serial.println(DC);
103
             while(DC < 230.0)                             //lower DC
     threshold value
105          {

107            VGC1 = VGC1 - 0.1;                          //Keep lowering the
      attenuation from VGC1 until DC value is greater than 230mV
               write_value = VGC1*(255/4.3);
109            analogWrite(digital6, write_value);
               delay(200);
111
               DC = DC_smoothing(readings, total, numReadings);
```

```
113
                //Serial.println(DC);
115             if(DC > 270.0)                                  //Upper thrwshold
    DC value
                {
117               VGC1 = VGC1 + 0.07;                           //Trial and error
    adjust value for DC to stay within 230 - 270 mV
                  write_value = VGC1*(255/4.3);
119               analogWrite(digital6, write_value);
                  delay(200);
121
                  DC = DC_smoothing(readings, total, numReadings);
123               }

125               if(VGC1 < 0.6)
                   goto bailout_for;                            //If the VGC1 sweep
     reaches its minimum, no relevant power spikes were found, bailout of VGC1
    sweep, move to next frequency
127
                }goto bailout_for;                              //When DC value is
    stable move to next frequency
129           }
            }bailout_for:
131       /*
           int fLO = 951;
133        int VGC1 = 21;
           int DC = 250;                                       //Test Values
135       */
           VGC1 = VGC1*100;                                     //Adjustment for
    VGC1 value, use it as an int
137
          ////////////////////////////////// Print / Send values to LoRa module
    ////////////////////////////////////////
139       Serial.write('i');
          delay(1);
141       Serial.write('\n');
          delay(1);
143       //Serial.println("fLO: ");
          //Serial.println(fLO);
145       send_char(fLO);

147       //Serial.println("VGC1: ");
          //Serial.println(VGC1);
149       send_char(VGC1);

151       //Serial.println("DC: ");
          //Serial.println(DC);
153       send_char(DC);

155       delay(50);
          ////////////////////////////////////////////////////////////////////
157       }
    }
159
    int MAX2112_config(int fLO)
161 {
      ////////////////////////////////////////////Editable Bits
```

```
    ////////////////////////////////////////////////
163
    //Registers BIT names
165 //N-Divider MSB
    int FRAC = 128;                 //Users must program to 1 upon powering up
    the device.
167 int N14_8 = 0;


169
    //Divider LSB
171 int N7_0 = round(fLO/27);        //Sets the most significant bits of the PLL
    integer-divide number (N). N can range from 19 to 251.


173
    //Charge-Pump
175 String CPMP = "00";             //Charge-pump minimum pulse width. Users must
     program to 00 upon powering up the device.

177 String CPLIN = "01";            //Controls Stringge-pump linearity. Users
    must program to 01 upon powering upthe device.

179 int REM = fLO%27;

181                                 //Sets the 4 most significant bits of the PLL
     fractional divide number
                                    //Default value is F = 194,180 decimal.
183 uint32_t F = 194180*REM/5;


185
    //XTAL buffer and reference divider
187
    String XD = "000";              /*000 = Divide by 1.
189                                  001 = Divide by 2.
                                     011 = Divide by 3.
191                                  100 = Divide by 4.
                                     101 through 110 = All divide values from 5
    (101) to 7 (110).
193                                  111 = Divide by 8.*/

195 String R = "00001";             /*Sets the PLL reference-divider (R) number.
    Users must program to 00001
                                     upon powering up the device.
197                                  00001 = Divide by 1; other values are not
    tested.*/

199 //PLL Register

201 int fVCO = 0;
    String D24 = "";
203 if (fLO <= 1125)                  //VCO divider setting.
    {
205     D24 = "1";                   //0 = Divide by 2. Use for LO frequencies >=
     1125MHz.
        fVCO = fLO*4;                //1 = Divide by 4. Use for LO frequencies <
    1125MHz.
207 }
    else
```

```
209     {
            D24 = "0";
211         fVCO = fLO*2;
        }
213
        //Serial.println()
215     String CPS = "1";                   /*Charge-pump current mode.
                                             0 = Charge-pump current controlled by ICP
        bit.
217                                          1 = Charge-pump current controlled by VCO
        autoselect (VAS).*/

219     String ICP = "0";                   /*Charge-pump current.
                                            0 = 600?A typical.
221                                         1 = 1200?A typical.*/

223     //VCO Register
        String VCO4_0 = "11001";            //Default value
225
        if (fVCO < f2_3)
227         VCO4_0 = "01010";    //VCO2
        else if (fVCO < f3_4)
229         VCO4_0 = "01011";    //VCO3
        else if (fVCO < f4_5)
231         VCO4_0 = "01100";    //VCO4
        else if (fVCO < f5_6)
233         VCO4_0 = "01101";    //VCO5
        else if (fVCO < f6_7)
235         VCO4_0 = "01110";    //VCO6
        else if (fVCO < f7_8)
237         VCO4_0 = "01111";    //VCO7
        else if (fVCO < f8_9)
239         VCO4_0 = "10000";    //VCO8
        else if (fVCO < f9_10)
241         VCO4_0 = "10001";    //VCO9
        else if (fVCO < f10_11)
243         VCO4_0 = "10010";    //VCO10
        else if (fVCO < f11_12)
245         VCO4_0 = "10011";    //VCO11
        else if (fVCO < f12_13)
247         VCO4_0 = "10100";    //VCO12
        else if (fVCO < f13_14)
249         VCO4_0 = "10101";    //VCO13
        else if (fVCO < f14_15)
251         VCO4_0 = "10110";    //VCO14
        else if (fVCO < f15_16)
253         VCO4_0 = "10111";    //VCO15
        else if (fVCO < f16_17)
255         VCO4_0 = "11000";    //VCO16
        else if (fVCO < f17_18)
257         VCO4_0 = "11001";    //VCO17
        else if (fVCO < f18_19)
259         VCO4_0 = "11010";    //VCO18
        else if (fVCO < f19_20)
261         VCO4_0 = "11011";    //VCO19
        else if (fVCO < f20_21)
263         VCO4_0 = "11100";    //VCO20
```

```
    else if (fVCO < f21_22)
        VCO4_0 = "11101";    //VCO21
    else if (fVCO < f22_23)
        VCO4_0 = "11110";    //VCO22
    else
        VCO4_0 = "11111";    //VCO23

//Controls which VCO is activated when using manual VCO programming mode.
//This also serves as the starting point for the VCO autoselection (VAS) mode.

    String VAS = "1"; /*VCO autoselection (VAS) circuit.
                        %0 = Disable VCO selection must be programmed through I2C.
                        %1 = Enable VCO selection controlled by autoselection
    circuit.*/

    String ADL = "0"; /*Enables or disables the VCO tuning voltage ADC latch when
     the VCO autoselect mode (VAS) is disabled.
                        %0 = Disables the ADC latch.
                        %1 = Latches the ADC value.*/

    String ADE = "0"; /*Enables or disables VCO tuning voltage ADC read when the
    VCO autoselect mode (VAS) is disabled.
                        %0 = Disables ADC read.
                        %1 = Enables ADC read.*/

    //Low Pass Filter
    double LPF_FREQ = 4E6;
    int LPF = round(((LPF_FREQ-4E6))/290E3+12); /*Sets the baseband lowpass
    filter 3dB corner frequency.
                                                %f-3dB = 4MHz + (LPF[7:0]dec - 12)
    x 290kHz.
                                                %Default value equates to f-3dB =
    22.27MHz typical.*/

    //Control Register
    String STBY = "0";  /*Software standby control.
                          %0 = Normal operation.
                          %1 = Disables the signal path and frequency synthesizer
    leaving only the 2-wire
                          %bus, crystal oscillator, XTALOUT buffer, and XTALOUT
    buffer divider active.*/

    String PWDN = "0";  /*Factory use only.
                          %0 = Normal operation;
                          %other value is not tested.*/

    int BBG = 15;       /*Baseband gain setting (1dB typical per step).
                          %0000 = Minimum gain (0dB, default).
                          %
                          %1111 = Maximum gain (15dB typical).*/

    //Shutdown Register

    String PLL = "0";   /*PLL enable.
                          %0 = Normal operation.
                          %1 = Shuts down the PLL. Value not tested.*/
```

```
313     String DIV = "0";    /*Divider enable.
                             %0 = Normal operation.
315                          %1 = Shuts down the divider. Value not tested.*/

317     String VCO = "0";    /*VCO enable.
                             %0 = Normal operation.
319                          %1 = Shuts down the VCO. Value not tested.*/

321     String BB = "0";     /*Baseband enable.
                             %0 = Normal operation.
323                          %1 = Shuts down the baseband. Value not tested.*/

325     String RFMIX = "0"; /*RF mixer enable.
                             %0 = Normal operation.
327                          %1 = Shuts down the RF mixer. Value not tested.*/

329     String RFVGA = "0"; /*RF VGA enable.
                             %0 = Normal operation.
331                          %1 = Shuts down the RF VGA. Value not tested.*/

333     String FE = "0";     /*Front-end enable.
                             %0 = Normal operation.
335                          %1 = Shuts down the front-end. Value not tested.  */

337     //Test Register
        String CPTST = "000";  /*Stringge-pump test modes.
339                             %000 = Normal operation (default).    */

341     String TURBO = "1";     /*Stringge-pump fast lock.
                               %Users must program to 1 after powering up the device`
        */
343
        String LDMUX = "000";  /*REFOUT output.
345                             %000 = Normal operation. Other values are not tested.
        */

347    //////////////////////////////////////// Write Registers
       ////////////////////////////////////////////////
       //N-Divider MSB
349    int MSB_REG = FRAC + N14_8;
       max2112_write(0x00, MSB_REG);
351    delay(5);

353    //PLL Register
       String PLLw = D24+CPS+ICP+"00000";
355    max2112_write(0x06, PLLw.toInt());
       delay(5);
357
       //N-Divider LSB
359    max2112_write(0x01, N7_0);
       delay(5);
361
       //Charge-Pump
363
       String F_bin = dec2bin(F,20);
365
       String F_bin_04 = F_bin.substring(0,4);
```

```
367
    String CP = CPMP+CPLIN+F_bin_04;
369 int CP_int = bin2dec(CP.toInt());
    max2112_write(0x02, CP_int);
371
    delay(5);
373
    //F Divider MSB
375 String F_bin_411= F_bin.substring(4,12);
    int F_bin_411_int = bin2dec(F_bin_411.toInt());
377 max2112_write(0x03, F_bin_411_int);
    delay(5);
379
    //F Divider LSB
381 String F_bin_1221= F_bin.substring(12,21);
    int F_bin_1221_int = bin2dec(F_bin_1221.toInt());
383 max2112_write(0x04, F_bin_1221_int);
    delay(5);
385
    //XTAL buffer and reference divider
387 String XTAL = XD+R;
    max2112_write(0x05, XTAL.toInt());
389 delay(5);

391 //VCO Register
    String VCOw = VCO4_0+VAS+ADL+ADE;
393
    int VCOw_int = bin2dec(VCOw.toInt());
395 max2112_write(0x07, VCOw_int);
    delay(5);
397
    //Low Pass Filter
399 max2112_write(0x08, LPF);
    delay(5);
401
    //Control Register
403 String BBGw = dec2bin(BBG,4);
    String CONTROL = STBY+"0"+PWDN+"0"+BBGw;
405 int CONTROL_int = bin2dec(CONTROL.toInt());
    max2112_write(0x09, CONTROL_int);
407 delay(5);

409 //Shutdown Register
    String SHUTDOWN = "0"+PLL+DIV+VCO+BB+RFMIX+RFVGA+FE;
411 max2112_write(0x0A, SHUTDOWN.toInt());
    delay(5);
413
    //Test Register
415 String TEST = CPTST+"0"+TURBO+LDMUX;
    int TEST_int = bin2dec(TEST.toInt());
417 max2112_write(0x0B, TEST_int);
    delay(5);
419 return LPF;
    //PrintStatusBytes(LPF);
421 }
  ////////////////////////////////////////////////////////////
423 ////////////////////////////////////////////End Config
```

```
  ////////////////////////////////////////////////////

//////////////////////////////////////Read Status byte registers and LPF
  ////////////////////////////////////////////
int PrintStatusBytes()
{
  //Read Status byte 1
  byte SByte1 = max2112_read(0x0C);     //Read Status Register 1
  String SB1 = dec2bin(SByte1,8);

  //Read each individual bit
  delay(5);
  int POR = bitRead(SByte1,7);
  delay(5);
  int VASA = bitRead(SByte1,6);
  delay(5);
  int VASE = bitRead(SByte1,5);
  delay(5);
  int LD = bitRead(SByte1,4);
  delay(5);

  //Read Status byte 2
  byte SByte2 = max2112_read(0x0D);
  String SB2 = dec2bin(SByte2,8);
  delay(5);

  //Print Bytes for debug
  /*Serial.println("Registos:");
  Serial.println(max2112_read(0x00),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x01),BIN); // Status byte 2
  delay(10);
  Serial.println(max2112_read(0x02),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x03),BIN); // Status byte 2
  delay(10);
  Serial.println(max2112_read(0x04),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x05),BIN); // Status byte 2
  delay(10);
  Serial.println(max2112_read(0x06),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x07),BIN); // Status byte 2
  delay(10);
  Serial.println(max2112_read(0x08),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x09),BIN); // Status byte 2
  delay(10);
  Serial.println(max2112_read(0x0A),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x0B),BIN); // Status byte 2
  delay(10);
  Serial.println(max2112_read(0x0C),BIN); // Status byte 1
  delay(10);
  Serial.println(max2112_read(0x0D),BIN); // Status byte 2
  delay(10);
  Serial.println("FIM");*/
```

```
479     String ADC1 = SB2.substring(5,8);       //Concatenate corresponding byte part
        String VCOSBR = SB2.substring(0,5);
481
483     //Print Status Byte 1
        Serial.print("POR: ");
485     Serial.println(POR);
        Serial.print("VASA: ");
487     Serial.println(VASA);
        Serial.print("VASE: ");
489     Serial.println(VASE);
        Serial.print("LD: ");
491     Serial.println(LD);

493     //Print Status Byte 2
        Serial.print("ADC: ");
495     Serial.println(ADC1);
        Serial.print("VCOSBR: ");
497     Serial.println(VCOSBR);

499     //Show rounded value of the LPF frequency (Register only takes integers which
         may lead to a real LPF frequency slightly off the intended one)
        /*double LPF_freq_round = 4E6 +(LPF-12)*290E3;
501     Serial.println("LPF freq: ");
        Serial.println(LPF_freq_round,DEC);
503     */

505     //Error conditions
        int errorcode = 1;              //Success
507     if(VASA == 0 || VASE == 0 || LD == 0)
        {
509       errorcode = 0;                //Error ocurred
        }
511
        return errorcode;
513   }
    ////////////////////////////////////////////////////////////////////////////
515
    ////////////////////////////////////// Convert decimal int number to binary
      ////////////////////////////////////////////////////////
517   String dec2bin(uint32_t value, int bits)
    {
519     int zeros = bits - String(value,BIN).length();
        String bin_str = "";
521     for (int i = 0; i < zeros; i++)
        {
523       bin_str += "0";
        }
525     bin_str += String(value, BIN);
        return bin_str;
527   }
    ////////////////////////////////////////////////////////////////////////////
529
    ////////////////////////////////////// Convert binary number to decimal
      ////////////////////////////////////////////////////////////
531   int bin2dec(uint32_t bin)
    {
```

```
533       int total  = 0;
          int potenc = 1;
535
          while(bin > 0) {
537           total += bin % 10 * potenc;
              bin    = bin / 10;
539           potenc = potenc * 2;
          }
541
          return total;
543   }
      ////////////////////////////////////////////////////////////////////////////////
545
      ////////////////////////////////////// Analog Read average calculator
        ////////////////////////////////////////////////////////////////
547   double DC_smoothing(int *readings, int total, int numReadings){
      int i;
549   int DC_an;
      for(i=0; i<numReadings; i++){
551
        // subtract the last reading:
553     //total = total - readings[i];
        // read from the sensor:
555     readings[i] = analogRead(analog1);
        delay(10);
557     // add the reading to the total:
        total = total + readings[i];
559     //Serial.println(total);
        }
561
        // calculate the average:
563     DC_an = total / numReadings;
        delay(1);        // delay in between reads for stability
565
        double DC = (5000*(double)DC_an)/1023;
567
       return DC;
569   }
      ////////////////////////////////////////////////////////////////////////////////
571
      ////////////////////////////////////////// Convert and Send array of characters
         to LoRa module ////////////////////////////////////////
573   void send_char(int num)
      {
575     int i;
        int divisor;
577
        String snum = String(num,DEC);
579     int slength = snum.length();

581     switch (slength)
        {
583       case 4:
          divisor = 1000;
585       break;

587       case 3:
```

```
        divisor = 100;
589       break;

591       case 2:
          divisor = 10;
593       break;

595       case 1:
          divisor = 1;
597       break;
      }
599
      for(i = 0; i < snum.length(); i++)
601   {

603     tip[i] = num / (divisor/((int)pow(10,i)));

605     delay(10);
        //tip[i] = num / 100;
607
        num = num - divisor/(pow(10,i))*tip[i];
609
        //num = num - 100*tip[i];
611
        Serial.write(tip[i]+'0');
613     delay(1);

615     if(i == snum.length()-1)
          Serial.write('\n');
617       delay(1);

619   }
    /*
621   for(i = 0; i = snum.length();i++)
      {
623     Serial.println(tip[i]);
      }
625   */
    }
627 ////////////////////////////////////////////////////////////////////

629 ////////////////////////////////////////////// I2C MAX2112 write register
      ////////////////////////////////////////////////////
    void max2112_write(byte wr_address, int data){
631   Wire.beginTransmission(MAX2112);
      Wire.write(wr_address);
633   Wire.write(data);
      Wire.endTransmission();
635 }
    ////////////////////////////////////////////////////////////////////
637
    ////////////////////////////////////////////// I2C MAX2112 read register
      ////////////////////////////////////////////////////
639 byte max2112_read(byte wr_address){
      Wire.beginTransmission(MAX2112);
641   Wire.write(wr_address);
      Wire.endTransmission();
```

```
643
    Wire.requestFrom(MAX2112,1);
645
    while(!Wire.available()){
647 }
    return Wire.read();
649 }
651 ////////////////////////////////////////////////////////////////////
```

Appendix B

IoT Modules

B.1 DRF TOOL and Modes of operation

The IoT modules DRF5150S and DRF4432S were made to work with each other, the only thing we have to do is to configure them with the same parameters with the DRF TOOL 5150 and they'll be good to go. This software has some tricky features, it must be run in administrator mode and the module must be correctly connected for it to work, otherwise an error message is displayed.

This tool offers two main working modes: Data transmission Mode ad Sensor Data mode. In this appendix we shall work through them.

B.1.1 Data Transmission Mode

This mode is the common data transmission mode, which is the one we use in the Spectrum Sensor to transmit and receive data. It does not require ID setup. To receive data from the MCU the 5th pin (TX) should be connected to GND through a 100 Ω resistor, and the 4th pin (RX) should be connected to TX on the MCU as shown in image 74.

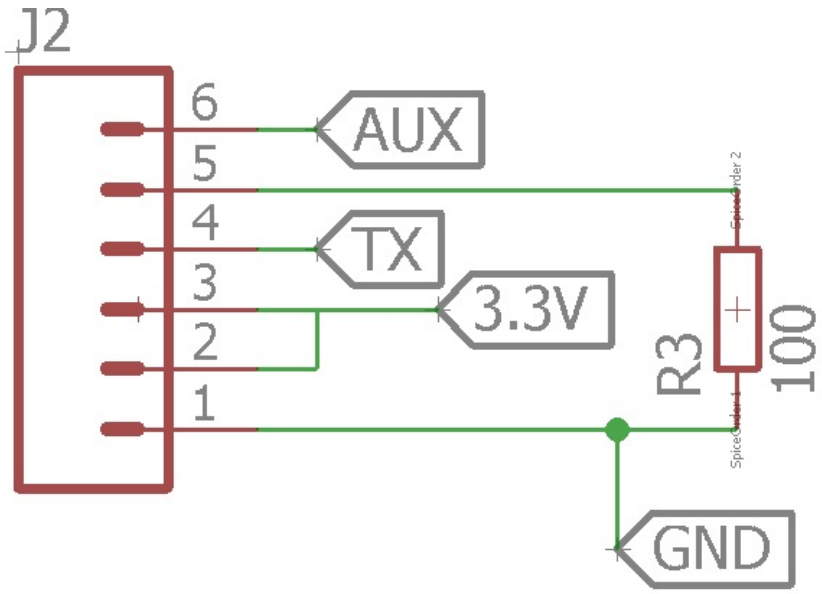


Figure 74: DRF5150S Wiring

Also very important regarding the DRF4432S receiver module is that when in receive mode, the SET-A pin must be connected to GND.

B.1.2 Sensor Data Mode

In this mode there, the configuration is made based on ID's, there is a Group ID and a Slave ID, basically if the transmitter DRF5150S has the same Group ID as the receiver DRF4432S, they will communicate with each other, the Slave ID is an identifier to which sensor we're receiving data from, with these two ID's it's possible to create sensor networks.

The data format of the transmitted data in this mode is: ID (group ID + Slave ID) + Data + Bat.

However the output data of the receiver has one extra byte RSSI, which indicates signal strength, as shown in the table below:

Data Format	Group ID	Slave ID	Data	Bat	RSSI
Length(byte)	1	1	2-4	1	1

Table 2: DRF4432S Output Data Format

The stated RSSI value for package loss is 0x40 at 50Kbps and 0x30 at 6.25kbps, although we've verified that the real value is a little above that.

The DRF TOOL offers three kinds of sensor settings. One for any type of **analogue sensors** such as potentiometers, another for **SHT1x and SHT2x** sensor from SENSIRION that integrate temperature and humidity, and finally the one that's of interest to this project which is the **DS18B20 Mode**, which was the used sensor to make coverage tests with the modules.

The configuration to these type of sensors is shown on figure 75.

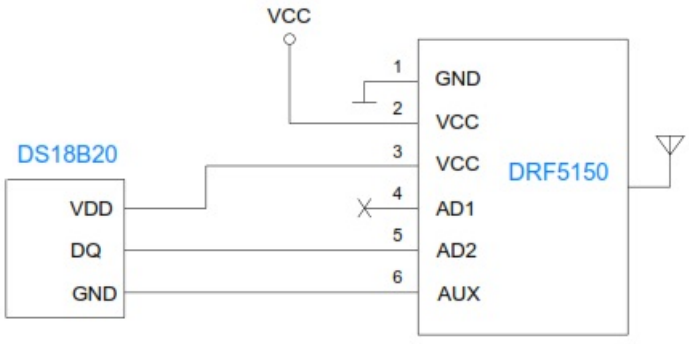


Figure 75: DRF5150S in DS18B20 Sensor Mode

This sensor is powered from 3.0 to 5.0 V and can measure temperature within -55 to +125 °C. It has a High and Low Resolution mode which impacts on the measurement time, and battery life.

B.2 MATLAB GUI Code for Temperature Sensor

This appendix portrays all the MATLAB code used to create the GUI for the IoT modules' test. The functions' headlines and Callbacks, in this case, were automatically generated by GUIDE, upon the creation of an interactive element on the interface, inside each callback we must only specify the function of the interactive element. The first function of the code is also completely generated by GUIDE since it's a required initialization code.

```
1       % Last Modified by GUIDE v2.5 07-Apr-2016 17:21:29

3 % Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
5 gui_State = struct('gui_Name',       mfilename, ...
                    'gui_Singleton',  gui_Singleton, ...
7                   'gui_OpeningFcn', @sensor_gui_2_OpeningFcn, ...
                    'gui_OutputFcn',  @sensor_gui_2_OutputFcn, ...
9                   'gui_LayoutFcn',  [] , ...
                    'gui_Callback',   []);
11 if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
13 end

15 if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
17 else
    gui_mainfcn(gui_State, varargin{:});
19 end
% End initialization code - DO NOT EDIT
21

23 % --- Executes just before sensor_gui_2 is made visible.
```

```
function sensor_gui_2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to sensor_gui_2 (see VARARGIN)
% Find a serial port object.
handles.Tsensor = instrfind('Type', 'serial', 'Port', 'COM3', 'Tag', '');

% Create the serial port object if it does not exist
% otherwise use the object that was found.
if isempty(handles.Tsensor)
    handles.Tsensor = serial('COM3');
else
    fclose(handles.Tsensor);
    handles.TSensor = handles.Tsensor(1);
end

fopen(handles.Tsensor);
% Firstly, obtain Slave ID for comparing
Byte_ind = fread(handles.Tsensor,[2,1]);
handles.ind = Byte_ind(2,1);
fclose(handles.Tsensor);


handles.y = [];
handles.x = [];
handles.z = [];

% Choose default command line output for sensor_gui_2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes sensor_gui_2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = sensor_gui_2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in temp_plot.
function temp_plot_Callback(hObject, eventdata, handles)
% hObject    handle to temp_plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tic %begin timer
```

```
for i = 1:1000
    fopen(handles.Tsensor);
    % Read bytes

    Byte = fread(handles.Tsensor,[12,1]); %Reads 6 (or 12) bytes according to the
     number of desired sensors

    %If the Slave ID is different from the one previously read it changes the
    positions of the 6 byte readings on matrix Byte

    if Byte(2,1) ~= handles.ind
    temp=zeros(6,1);
    for j=1:6
    temp(j,1) = Byte(j,1);
    Byte(j,1) = Byte(j+6,1);
    Byte(j+6,1)= temp(j,1);
    end
    end

     fclose(handles.Tsensor);

    % DS18B20 conversion
    tempC(i) = (Byte(4,1)*256+Byte(3,1))/16;
    BattV(i) = (Byte(5,1)+200)/100;
    %tempF = ((tempC*9)/5)+32   %Farehneit value
    rssi(i) = Byte(6,1);

    %Refreshes static text int the GUI
    set(handles.slaveid,'string',Byte(2,1));
    set(handles.temp,'string',tempC(i));
    set(handles.rssi,'string',rssi(i));
    set(handles.batt,'string',BattV(i));
    drawnow();


    S1 = sprintf('Slave ID %d', Byte(2,1));
    handles.x=[handles.x toc];
    handles.y=[handles.y tempC(i)];

    %If there's more than one sensor
   if Byte(2,1) ~= Byte(8,1)
        tempC2(i) = (Byte(10,1)*256+Byte(9,1))/16;
        BattV2(i) = (Byte(11,1)+200)/100;
        %tempF = ((tempC*9)/5)+32
        rssi2(i) = Byte(12,1);

        %Refreshes static text int the GUI
        set(handles.slaveid2,'string',Byte(8,1));
        set(handles.temp2,'string',tempC2(i));
        set(handles.rssi2,'string',rssi2(i));
        set(handles.bat2,'string',BattV2(i));
        drawnow();

        %Real time plot of both sensors
        S2 = sprintf('Slave ID %d', Byte(8,1));
        handles.z=[handles.z tempC2(i)];
        plot(handles.x, handles.y, handles.x, handles.z, 'r');
```

```
        title('Temperatura');
137     xlabel('Tempo');
        ylabel('Temperatura');
139     legend(S1,S2)
        drawnow();
141 else
    %Real time plot of one sensor
143     plot(handles.x, handles.y);
        title('Temperatura');
145     xlabel('Tempo');
        ylabel('Temperatura');
147     legend(S1)
        drawnow();
149 end


151
end
153


155 % --- Executes on button press in rssi_plot.
function rssi_plot_Callback(hObject, eventdata, handles)
157 % hObject    handle to rssi_plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
159 % handles    structure with handles and user data (see GUIDATA)
tic %begin timer
161
for i = 1:1000
163     fopen(handles.Tsensor);
    % Read bytes
165
    Byte = fread(handles.Tsensor,[12,1]); %Reads 6 (or 12) bytes according to the
     number of desired sensors
167
    %If the Slave ID is different from the one previously read it changes the
    positions of the 6 byte readings on matrix Byte
169 if Byte(2,1) ~= handles.ind
    temp=zeros(6,1);
171 for j=1:6
    temp(j,1) = Byte(j,1);
173 Byte(j,1) = Byte(j+6,1);
    Byte(j+6,1)= temp(j,1);
175 end
    end
177
     fclose(handles.Tsensor);
179
    % DS18B20 conversion
181 tempC(i) = (Byte(4,1)*256+Byte(3,1))/16;
    BattV(i) = (Byte(5,1)+200)/100;
183 %tempF = ((tempC*9)/5)+32
    rssi(i) = Byte(6,1);
185
    %Refreshes static text int the GUI
187 set(handles.slaveid,'string',Byte(2,1));
    set(handles.temp,'string',tempC(i));
189 set(handles.rssi,'string',rssi(i));
    set(handles.batt,'string',BattV(i));
```

```
191     drawnow();

193
        S1 = sprintf('Slave ID %d', Byte(2,1));
195     handles.x=[handles.x toc];
        handles.y=[handles.y rssi(i)];
197
       %If there's more than one sensor
199    if Byte(2,1) ~= Byte(8,1)
            tempC2(i) = (Byte(10,1)*256+Byte(9,1))/16;
201         BattV2(i) = (Byte(11,1)+200)/100;
           %tempF = ((tempC*9)/5)+32
203         rssi2(i) = Byte(12,1);

205         %Refreshes static text int the GUI
            set(handles.slaveid2,'string',Byte(8,1));
207         set(handles.temp2,'string',tempC2(i));
            set(handles.rssi2,'string',rssi2(i));
209         set(handles.bat2,'string',BattV2(i));
            drawnow();
211
            %Real time plot of both sensors
213         S2 = sprintf('Slave ID %d', Byte(8,1));
            handles.z=[handles.z rssi2(i)];
215         plot(handles.x, handles.y, handles.x, handles.z, 'r');
            title('RSSI');
217         xlabel('Tempo');
            ylabel('RSSI');
219         legend(S1,S2)
            drawnow();
221    else
           %Real time plot of one sensor
223        plot(handles.x, handles.y);
           title('RSSI');
225        xlabel('Tempo');
           ylabel('RSSI');
227        legend(S1)
           drawnow();
229    end


231
end
233
```

B.3 MATLAB Data Receive Code

```
clear;
clc;
close all;

%%
%%%%%%%%%%%%%% Data arrays %%%%%%%%%%%%%%%%%%%%%%
fLO = [];
DC = [];
VGC1 = [];
Pin_mw = [];
Pin_dbm = [];
Pin_real = [];

%%
% Find a serial port object.
Tsensor = instrfind('Type', 'serial', 'Port', 'COM3', 'Tag', '');

% Create the serial port object if it does not exist otherwise use the object
    that was found.
if isempty(Tsensor)
    Tsensor = serial('COM3');
else
    fclose(Tsensor);
    TSensor = Tsensor(1);
end
set(Tsensor,'Timeout',30); %Allow for the script to wait enough time for a byte

%%
% Connect to instrument object, obj1.
fopen(Tsensor);

%%%%%%%%%%%%%%%%%%% Read Loop %%%%%%%%%%%%%%%%%%%%%%
while(1)
    for j = 1:100

        i = fscanf(Tsensor,'%c');
        while ( i ~= 'i')
            i = fscanf(Tsensor,'%c');
        end

        fLO = [fLO fscanf(Tsensor)];
        VGC1 = [VGC1 fscanf(Tsensor)];
        DC = [DC fscanf(Tsensor)];

  %Convert string values to numbers
        fLO_j = str2num(fLO);
        VGC1_j = str2num(VGC1);
        DC_j = str2num(DC);

   Poutf = 6.2691*log(DC_j) - 39.542;
        Gain = (VGC1_n - 0.7)/0.0247;
        Pin = polyval(p_in_out, (Poutf+Gain));    %Power conversions

```

```
        for k = 1 : size(Pin,1)          %Define everything above 0dBm and below
    -100dBm as noise (-100dBm)
54          if Pin(k,1) < -100 || Pin(k,1) > 0
                Pin(k,1) = -100;
56          end
        end
58      plot(fLO_j, Pin);
        axis([-inf,+inf,-110,0])         %Define maximum plot limits for better
    view
60
        drawnow();              %Plot in real time
62     % save('test_results.txt', 'fLO_j', 'VGC1_n', 'DC_j', 'Pin');
64
    end
66 end
```